はじめての HPKI
～実装の手引き～

分冊　サンプルコード

2021 年 3 月

一般財団法人医療情報システム開発センター

はじめての HPKI
～実装の手引き～

分冊　サンプルコード目次

サンプルコード

・ソースコード＜PKCS#11 インタフェース利用＞

・HPKISignVerifySampleP11.c

```
/*-------------------------------------------------------------------------------*/
/* FILE NAME :    HPKISignVerifySampleP11.c                                      */
/* VERSION    :    1.0                                                           */
/* DATE    :    2020/11/21                                                       */
/*-------------------------------------------------------------------------------*/

/*===============================================================*/
/*                                INCLUDE                                        */
/*===============================================================*/

#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include "pkcs11.h"

#include <malloc.h>
#include <winbase.h>

#include "openssl/bio.h"
#include "openssl/evp.h"
#include "openssl/x509.h"
#include "openssl/rsa.h"
#include "openssl/engine.h"
#include "openssl/objects.h"
#include "openssl/sha.h"
#include "openssl/err.h"

/*===============================================================*/
/*                                DEFINITION                                     */
/*===============================================================*/

typedef CK_RV(*C_GetFunctionListFuncPtr)(CK_FUNCTION_LIST_PTR_PTR ppFu
nctionList);

#define RETURN_SUCCESS              0
#define RETURN_FAILURE              1
#define NUM_SLOT                    4
#define FIND_TEMPLATE_COUNT         3
#define ATTRIBUTE_TEMPLATE_COUNT 1
#define BUFFER_SIZE                 512

/*===============================================================*/
/*                DEFINITIOIN OF PRIVATE FUNCTION                */
/*===============================================================*/

/* プログラムの実行形式 */
void PrintUsage() {
```

```
    printf("Usage : HPKISignVerifySampleP11 <PKCS#11 Library_Type> <Pin> ¥n
");
    printf("PKCS#11 Library_Type¥t: auth | sign¥n");
    printf("Pin¥t¥t¥t: HPKICardPin¥n");
}

/* 各データ(証明書データ、署名データ)を 16 進数表示で標準出力 */
void printHex(const BYTE* data, DWORD len, BOOL limit)
{
    DWORD displen;

    if (limit && len > 256)
        displen = 256;
    else
        displen = len;
    for (DWORD i = 0; i < displen; i++) {
        printf("%02x", *(data + i));
        if (i % 16 == 15 || i == displen - 1)
            printf("¥n");
        else
            printf(" ");
    }
    if (displen < len)
        printf("...¥n");
}
/* バイナリデータをファイル出力*/
int fileout(const char* outf, const BYTE* data, DWORD len)
{
    FILE *fp;
    if (fopen_s(&fp,outf, "wb") != 0) {  /* ファイルのオープン */
        printf("file open error!!¥n");
        return 1;
    }
    fwrite(data, len, 1, fp);
    fclose(fp);
    return 0;
}

/*====================================================================*/
/*                      Start of main()                               */
/*====================================================================*/

int main(int argc, char **argv) {

    const char                  *pPkcs11LibraryName;

    CK_UTF8CHAR_PTR             pUserPin;
    CK_ULONG                    userPinLen;
    CK_BYTE_PTR                 pMessage;
    CK_ULONG                    messageLen;
```

```
    CK_BBOOL                    loadLibraryFlag = FALSE;
    CK_BBOOL                    initializeFlag = FALSE;
    CK_BBOOL                    openSessionFlag = FALSE;
    CK_BBOOL                    loginFlag = FALSE;
    CK_BBOOL                    findObjectsFlag = FALSE;

    HINSTANCE                    instance = NULL;
    FARPROC                     procAddress = NULL;
    BOOL                        freeLibraryRc;

    CK_RV                       rv;
    CK_FUNCTION_LIST_PTR        pFunctionList = NULL;
    CK_ULONG                     slotCount;
    CK_SLOT_ID                  slotID[NUM_SLOT];
    CK_SESSION_HANDLE            sessionHandle;
    CK_TOKEN_INFO               tokenInfo;

    CK_OBJECT_CLASS              objectClass;
    CK_BBOOL                    ckTrue = TRUE;
    CK_ATTRIBUTE                 findTemplate[FIND_TEMPLATE_COUNT];
    CK_ULONG                    findTemplateCount = FIND_TEMPLATE_COUNT;
    CK_OBJECT_HANDLE             privateKeyObjectHandle;
    CK_ULONG                    foundObjectCount;
    CK_UTF8CHAR_PTR              pLabel;
    CK_ULONG                    labelLen;

    CK_ATTRIBUTE                 attributeTemplate[ATTRIBUTE_TEMPLATE_COU
NT];
    CK_ULONG                    attributeTemplateCount = ATTRIBUTE_TEMPLAT
E_COUNT;
    CK_OBJECT_HANDLE             certificateObjectHandle;

    CK_MECHANISM                 signatureMechanism;
    CK_BYTE                     signature[BUFFER_SIZE];
    CK_ULONG                     signatureLen = BUFFER_SIZE;

    unsigned char               *pCertificate = NULL;
    unsigned long               certificateLen = 0;

    BIO          *pBioCertificate = NULL;
    X509                        *pX509Certificate = NULL;
    EVP_PKEY                     *pEvpPublicKey = NULL;
    RSA              *pRsaPublicKey = NULL;
    unsigned char                hash[SHA256_DIGEST_LENGTH];
    int                         rc;

    typedef struct X509_sig_st_S {
        X509_ALGOR *algor;
        ASN1_OCTET_STRING *digest;
    }X509_SIG_S;
    X509_SIG_S sig;
    X509_ALGOR algor;
```

```
ASN1_TYPE parameter;
ASN1_OCTET_STRING digest;
uint8_t *der = NULL;
int len;

/*-------------------------------------------------------------------------------*/
/*   引数のチェック                                                              */
/*-------------------------------------------------------------------------------*/

if (argc != 3)
{
    printf("実行形式が正しくありません。¥n");
    PrintUsage();
    return(RETURN_FAILURE);
}

/* HPKI P11 ライブラリの選択*/
switch (*(argv[1])) {
    /*電子認証用*/
case 'a':
case 'A':
    pPkcs11LibraryName = "HpkiAuthP11_MPKCS11H.dll";
    break;
    /*電子署名用*/
case 's':
case 'S':
    pPkcs11LibraryName = "HpkiSigP11_MPKCS11H.dll";
    break;
default:
    printf("実行パラメータ<PKCS#11 Library_Type>が正しくありません。¥n");
    PrintUsage();
    exit(EXIT_FAILURE);
    break;
}

/* PIN */
pUserPin = (CK_UTF8CHAR_PTR)*(argv + 2);
userPinLen = (CK_ULONG)strlen((const char *)*(argv + 2));

/*-------------------------------------------------------------------------------*/
/*          初期処理                                                             */
/*-------------------------------------------------------------------------------*/

/*   (1) 選択された HPKI P11 ライブラリのロード  */
instance = LoadLibrary((LPCTSTR)pPkcs11LibraryName);
if (instance == NULL)
{
    printf("Error in %s() : %08x¥n", "LoadLibrary", GetLastError());
    goto m_end;
}
```

```
loadLibraryFlag = TRUE;

/*   (2) 動作させるための DLL 内の関数を取得 */
procAddress = GetProcAddress(instance, (LPCTSTR)"C_GetFunctionList");
if (procAddress == NULL)
{
    printf("Error in %s() : %08x¥n", "GetProcAddress", GetLastError());
    goto m_end;
}

/*   (3) 関数のポインタリストの取得   */
rv = ((C_GetFunctionListFuncPtr)(procAddress))(&pFunctionList);
if (rv != CKR_OK)
{
    printf("Error in %s() : %08x¥n", "C_GetFunctionList", rv);
    goto m_end;
}

/*   (4) HPKI P11 ライブラリを初期化する */
rv = (pFunctionList->C_Initialize)(NULL_PTR);
if (rv != CKR_OK)
{
    printf("Error in %s() : %08x¥n", "C_Initialize", rv);
    goto m_end;
}

initializeFlag = TRUE;

/*-------------------------------------------------------------------------------*/
/*          証明書取得に対する初期処理                                           */
/*-------------------------------------------------------------------------------*/

/*   (5) スロットリストを取得する(メモリ割当) */
rv = (pFunctionList->C_GetSlotList)(TRUE, NULL_PTR, &slotCount);
if (rv != CKR_OK)
{
    printf("Error in %s(1) : %08x¥n", "C_GetSlotList", rv);
    goto m_end;
}

if (slotCount < 1)
{
    printf("Invalid Slot Count¥n");
    goto m_end;
}

/*   (6) スロットリストを取得する */
rv = (pFunctionList->C_GetSlotList)(TRUE, slotID, &slotCount);
if (rv != CKR_OK)
{
    printf("Error in %s(2) : %08x¥n", "C_GetSlotList", rv);
```

```
        goto m_end;
    }

    /*  (7) 先頭のスロットに対するセッションを確立する */
    rv = (pFunctionList->C_OpenSession)(slotID[0], CKF_SERIAL_SESSION, NULL
_PTR, NULL_PTR, &sessionHandle);
    if (rv != CKR_OK)
    {
        printf("Error in %s() : %08x¥n", "C_OpenSession", rv);
        goto m_end;
    }

    openSessionFlag = TRUE;

    /*-------------------------------------------------------------------------*/
    /*    証明書取得処理                                                       */
    /*-------------------------------------------------------------------------*/

    printf("証明書取得処理 開始¥n");

    /* 検索条件の設定 */
    objectClass = CKO_CERTIFICATE;
    pLabel = "HPKI END ENTITY CERTIFICATE";
    labelLen = (CK_ULONG)strlen((const char *)pLabel);

    findTemplate[0].type = CKA_CLASS;
    findTemplate[0].pValue = &objectClass;
    findTemplate[0].ulValueLen = sizeof(objectClass);
    findTemplate[1].type = CKA_TOKEN;
    findTemplate[1].pValue = &ckTrue;
    findTemplate[1].ulValueLen = sizeof(ckTrue);
    findTemplate[2].type = CKA_LABEL;
    findTemplate[2].pValue = pLabel;
    findTemplate[2].ulValueLen = labelLen;

    /*  (8) オブジェクトの検索の初期化 */
    rv = (pFunctionList->C_FindObjectsInit)(sessionHandle, findTemplate, findTemp
lateCount);
    if (rv != CKR_OK)
    {
        printf("Error in %s() : %08x¥n", "C_FindObjectsInit_cert", rv);
        goto m_end;
    }

    findObjectsFlag = TRUE;

    /*  (9) オブジェクトの検索を行い、先頭の証明書を指定する */
    rv = (pFunctionList->C_FindObjects)(sessionHandle, &certificateObjectHandle,
1, &foundObjectCount);
    if (rv != CKR_OK)
    {
```

```
        printf("Error in %s() : %08x¥n", "C_FindObjects_cert", rv);
        goto m_end;
    }

    if (foundObjectCount != 1)
    {
        printf("Certificate Not Found¥n");
        goto m_end;
    }

    /* 属性の設定 */
    attributeTemplate[0].type = CKA_VALUE;
    attributeTemplate[0].pValue = NULL_PTR;
    attributeTemplate[0].ulValueLen = 0;

    /*  (10) オブジェクトの属性値を取得し、証明書のサイズを取得 */
    rv = (pFunctionList->C_GetAttributeValue)(sessionHandle, certificateObjectHan
dle, attributeTemplate, attributeTemplateCount);
    if (rv != CKR_OK)
    {
        printf("Error in %s(1) : %08x¥n", "C_GetAttributeValue", rv);
        goto m_end;
    }

    if (attributeTemplate[0].ulValueLen == 0)
    {
        printf("Invalid Certificate Size¥n");
        goto m_end;
    }

    /*  (11) メモリの確保を行い、証明書を取得する */
    certificateLen = attributeTemplate[0].ulValueLen;
    pCertificate = (unsigned char *)malloc((size_t)certificateLen);

    if (pCertificate == NULL)
    {
        printf("Memory Allocation Failure¥n");
        goto m_end;
    }

    /* 証明書の取得 */
    attributeTemplate[0].pValue = (CK_BYTE_PTR)pCertificate;

    rv = (pFunctionList->C_GetAttributeValue)(sessionHandle, certificateObjectHan
dle, attributeTemplate, attributeTemplateCount);
    if (rv != CKR_OK)
    {
        printf("Error in %s(2) : %08x¥n", "C_GetAttributeValue", rv);
        goto m_end;
    }
```

```
    /*   (12) オブジェクトの検索を終了する */
    findObjectsFlag = FALSE;

    rv = (pFunctionList->C_FindObjectsFinal)(sessionHandle);
    if (rv != CKR_OK)
    {
        printf("Error in %s() : %08x¥n", "C_FindObjectsFinal_cert", rv);
        goto m_end;
    }

    /* 証明書データを表示 */
    printf("Certificate:¥n");
    printHex(pCertificate, certificateLen, TRUE);
    /* 証明書データをファイル出力 */
    //rv = fileout("c:¥¥temp¥¥EEcert.der", pCertificate, certificateLen);
    if (rv != 0)
    {
        printf("Cetificate File Out Failure¥n");
        goto m_end;
    }
    printf("証明書取得処理 成功: %s¥n¥n", pLabel);

    /*-------------------------------------------------------------------------*/
    /*   署名処理                                                              */
    /*-------------------------------------------------------------------------*/

    printf("署名処理 開始¥n");

    /*   (13) 先頭のスロットのトークン情報を取得する */
    rv = (pFunctionList->C_GetTokenInfo)(slotID[0], &tokenInfo);
    if (rv != CKR_OK)
    {
        printf("Error in %s() : %08x¥n", "C_GetTokenInfo", rv);
        goto m_end;
    }

    /*   (14) 暗号トークンにログインする */
    if ((tokenInfo.flags & CKF_LOGIN_REQUIRED) != FALSE) {
        rv = (pFunctionList->C_Login)(sessionHandle, CKU_USER, pUserPin, user
PinLen);
        if (rv != CKR_OK)
        {
            printf("Error in %s() : %08x¥n", "C_Login", rv);
            goto m_end;
        }

        loginFlag = TRUE;
    }

    /*   (15) 暗号トークン中の署名生成用秘密鍵の検索条件を設定 */
    objectClass = CKO_PRIVATE_KEY;
```

```
    pLabel = "Private key of HPKI";
    labelLen = (CK_ULONG)strlen((const char *)pLabel);

    findTemplate[0].type = CKA_CLASS;
    findTemplate[0].pValue = &objectClass;
    findTemplate[0].ulValueLen = sizeof(objectClass);
    findTemplate[1].type = CKA_TOKEN;
    findTemplate[1].pValue = &ckTrue;
    findTemplate[1].ulValueLen = sizeof(ckTrue);
    findTemplate[2].type = CKA_LABEL;
    findTemplate[2].pValue = pLabel;
    findTemplate[2].ulValueLen = labelLen;

    /*  (16) オブジェクトの検索の初期化 */
    rv = (pFunctionList->C_FindObjectsInit)(sessionHandle, findTemplate, findTemp
lateCount);
    if (rv != CKR_OK)
    {
        printf("Error in %s() : %08x¥n", "C_FindObjectsInit_key", rv);
        goto m_end;
    }

    findObjectsFlag = TRUE;

    /*  (17) オブジェクトの検索を行い、先頭の秘密鍵を取得する */
    rv = (pFunctionList->C_FindObjects)(sessionHandle, &privateKeyObjectHandle,
1, &foundObjectCount);
    if (rv != CKR_OK)
    {
        printf("Error in %s() : %08x¥n", "C_FindObjects_key", rv);
        goto m_end;
    }

    if (foundObjectCount != 1)
    {
        printf("Private Key Not Found¥n");
        goto m_end;
    }

    /*  (18) オブジェクトの検索を終了する */
    findObjectsFlag = FALSE;
    rv = (pFunctionList->C_FindObjectsFinal)(sessionHandle);
    if (rv != CKR_OK)
    {
        printf("Error in %s() : %08x¥n", "C_FindObjectsFinal_key", rv);
        goto m_end;
    }

    /* 署名処理の初期化 */

    /*  (19) 署名を行うデータの生成 */
```

```
    pMessage = "hello-world";
    messageLen = (CK_ULONG)strlen((const char *)pMessage);

    /*  (20) 署名対象メッセージのハッシュ値を生成 */
    SHA256(pMessage, messageLen, hash);
    printf("SHA256 HASH:¥n");
    printHex(hash, SHA256_DIGEST_LENGTH, FALSE);
    printf("¥n");
    //rv = fileout("c:¥¥temp¥¥hash.dat", hash, SHA256_DIGEST_LENGTH);

    /*  (21) ハッシュ値の DigestInfo 生成 */
    sig.algor = &algor;
    sig.algor->algorithm = OBJ_nid2obj(NID_sha256);
    parameter.type = V_ASN1_NULL;
    parameter.value.ptr = NULL;
    sig.algor->parameter = &parameter;
    sig.digest = &digest;
    sig.digest->data = (unsigned char *)hash;
    sig.digest->length = SHA256_DIGEST_LENGTH;
    len = i2d_X509_SIG((X509_SIG *)&sig, &der);

    /*  (22) 署名メカニズムの設定 */
    signatureMechanism.mechanism = CKM_RSA_PKCS;
    signatureMechanism.pParameter = NULL_PTR;
    signatureMechanism.ulParameterLen = 0;

    /*  (23) 署名処理の初期化 */
    rv = (pFunctionList->C_SignInit)(sessionHandle, &signatureMechanism, private
KeyObjectHandle);
    if (rv != CKR_OK)
    {
        printf("Error in %s() : %08x¥n", "C_SignInit", rv);
        goto m_end;
    }

    /*  (24) データに署名を行う */
    rv = (pFunctionList->C_Sign)(sessionHandle, der, len, signature, &signatureLe
n);
    if (rv != CKR_OK)
    {
        printf("Error in %s() : %08x¥n", "C_Sign", rv);
        goto m_end;
    }

    /* 署名データを表示 */
    printf("Signature:¥n");
    printHex(signature, signatureLen, FALSE);
    printf("署名処理 成功¥n¥n");
    //rv = fileout("c:¥¥temp¥¥sig.dat", signature, signatureLen);
```

```
    /*+++++++++++++++++++++++++++++++++++++++++++++*/
    /*   ※署名値の確認          Start                */
    /*+++++++++++++++++++++++++++++++++++++++++++++*/

    printf("検証処理 開始¥n");

    struct rsa_st *pRsaPlivateKey = NULL;

    /*   (25) メモリの確保 */
    pBioCertificate = BIO_new_mem_buf((void *)pCertificate, (int)certificateLen);
    if (pBioCertificate == NULL)
    {
        printf("Error in %s() : NULL¥n", "BIO_new_mem_buf");
        goto m_end;
    }

    /*   (26) 公開鍵証明書の復号 */
    pX509Certificate = d2i_X509_bio(pBioCertificate, NULL);
    if (pX509Certificate == NULL)
    {
        printf("Error in %s() : NULL¥n", "d2i_X509_bio");
        goto m_end;
    }

    /*   (27) 公開鍵の取得 */
    pEvpPublicKey = X509_get_pubkey(pX509Certificate);
    if (pEvpPublicKey == NULL)
    {
        printf("Error in %s() : NULL¥n", "X509_get_pubkey");
        goto m_end;
    }

    pRsaPublicKey = EVP_PKEY_get1_RSA(pEvpPublicKey);
    if (pRsaPublicKey == NULL)
    {
        printf("Error in %s() : NULL¥n", "EVP_PKEY_get1_RSA");
        goto m_end;
    }

    /*   (28) 署名検証 */
    rc = RSA_verify(NID_sha256, hash, SHA256_DIGEST_LENGTH, signature, sig
natureLen, pRsaPublicKey);
    if (rc != 1)
    {
        printf("検証処理 失敗¥n");
        printf("ErrorCode : %08x¥n", ERR_get_error()); //04091068
        goto m_end;
    }

    printf("検証処理 成功¥n");
```

```
/*   (29) 公開鍵の解放   */
RSA_free(pRsaPublicKey);
pRsaPublicKey = NULL;

EVP_PKEY_free(pEvpPublicKey);
pEvpPublicKey = NULL;

/*   (30) 公開鍵証明書の解放   */
X509_free(pX509Certificate);
pX509Certificate = NULL;

/*   (31) メモリ領域の解放   */
BIO_free(pBioCertificate);
pBioCertificate = NULL;

/*+++++++++++++++++++++++++++++++++++++++++++++*/
/*   ※署名値の確認          End                 */
/*+++++++++++++++++++++++++++++++++++++++++++++*/

/*-----------------------------------------------------------------------------*/
/*     終了処理                                                                */
/*-----------------------------------------------------------------------------*/

/*   (32) 証明書の解放 */
free(pCertificate);
pCertificate = NULL;

/*   (33) 暗号トークンからのログアウト */
loginFlag = FALSE;

rv = (pFunctionList->C_Logout)(sessionHandle);
if (rv != CKR_OK)
{
     printf("Error in %s(1) : %08x¥n", "C_Logout(", rv);
     goto m_end;
}

/*   (34) セッションのクローズ */
openSessionFlag = FALSE;

rv = (pFunctionList->C_CloseSession)(sessionHandle);
if (rv != CKR_OK)
{
     printf("Error in %s(1) : %08x¥n", "C_CloseSession", rv);
     goto m_end;
}

/*   (35) HPKI P11 ライブラリの終了 */
initializeFlag = FALSE;

rv = (pFunctionList->C_Finalize)(NULL_PTR);
```

```
    if (rv != CKR_OK)
    {
        printf("Error in %s(1) : %08x¥n", "C_Finalize", rv);
        goto m_end;
    }

    /*   (36) HPKI P11 ライブラリの解放 */
    loadLibraryFlag = FALSE;

    freeLibraryRc = FreeLibrary(instance);
    if (freeLibraryRc == 0)
    {
        printf("Error in %s(1) : %08x¥n", "FreeLibrary", GetLastError());
        goto m_end;
    }

    return(RETURN_SUCCESS);

m_end:


    /*-------------------------------------------------------------------------------*/
    /*   エラー発生時の終了処理                                                      */
    /*-------------------------------------------------------------------------------*/

    /*-------------------------------------------------------------------------------*/
    /* 公開鍵の解放(エラー発生時)                                                    */
    /*-------------------------------------------------------------------------------*/

    if (pRsaPublicKey != NULL)
    {
        RSA_free(pRsaPublicKey);
        pRsaPublicKey = NULL;
    }

    if (pEvpPublicKey != NULL)
    {
        EVP_PKEY_free(pEvpPublicKey);
        pEvpPublicKey = NULL;
    }

    /*-------------------------------------------------------------------------------*/
    /* 公開鍵証明書の解放(エラー発生時)                                              */
    /*-------------------------------------------------------------------------------*/

    if (pX509Certificate != NULL)
    {
        X509_free(pX509Certificate);
        pX509Certificate = NULL;
    }
```

```
/*-------------------------------------------------------------------------------*/
/* メモリ領域の解放(エラー発生時)                                                */
/*-------------------------------------------------------------------------------*/

if (pBioCertificate != NULL)
{
    BIO_free(pBioCertificate);
    pBioCertificate = NULL;

}

/*-------------------------------------------------------------------------------*/
/*   証明書の解放(エラー発生時)                                                  */
/*-------------------------------------------------------------------------------*/

if (pCertificate != NULL)
{
    free(pCertificate);
    pCertificate = NULL;
}

/*-------------------------------------------------------------------------------*/
/*   暗号トークンからのログアウト(エラー発生時)                                  */
/*-------------------------------------------------------------------------------*/

if (loginFlag == TRUE)
{
    loginFlag = FALSE;

    rv = (pFunctionList->C_Logout)(sessionHandle);
    if (rv != CKR_OK)
    {
        printf("Error in %s(2) : %08x¥n", "C_Logout", rv);
    }
}

/*-------------------------------------------------------------------------------*/
/*   セッションのクローズ(エラー発生時)                                          */
/*-------------------------------------------------------------------------------*/

if (openSessionFlag == TRUE)
{
    openSessionFlag = FALSE;

    rv = (pFunctionList->C_CloseSession)(sessionHandle);
    if (rv != CKR_OK)
    {
        printf("Error in %s(2) : %08x¥n", "C_CloseSession", rv);
    }
}
```

```
    /*-------------------------------------------------------------------------*/
    /*    HPKI P11 ライブラリの終了(エラー発生時)                              */
    /*-------------------------------------------------------------------------*/

    if (initializeFlag == TRUE)
    {
        initializeFlag = FALSE;

        rv = (pFunctionList->C_Finalize)(NULL_PTR);
        if (rv != CKR_OK)
        {
            printf("Error in %s(2) : %08x¥n", "C_Finalize", rv);
        }
    }

    /*-------------------------------------------------------------------------*/
    /*    HPKI P11 ライブラリの解放(エラー発生時)                              */
    /*-------------------------------------------------------------------------*/

    if (loadLibraryFlag == TRUE)
    {
        loadLibraryFlag = FALSE;

        freeLibraryRc = FreeLibrary(instance);
        if (freeLibraryRc == 0)
        {
            printf("Error in %s(2) : %08x¥n", "FreeLibrary", GetLastError());
        }
    }

    return(RETURN_FAILURE);
}

/*=========================================================================*/
/*            End of main()                                                */
/*=========================================================================*/
```

pkcs11.h

```
/* pkcs11.h
   Copyright 2006, 2007 g10 Code GmbH
   Copyright 2006 Andreas Jellinghaus

   This file is free software; as a special exception the author gives
   unlimited permission to copy and/or distribute it, with or without
   modifications, as long as this notice is preserved.
```

This file is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY, to the extent permitted by law; without even
the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICUL
AR
PURPOSE. */

/* Please submit changes back to the Scute project at
http://www.scute.org/ (or send them to marcus@g10code.com), so that
they can be picked up by other projects from there as well. */

/* This file is a modified implementation of the PKCS #11 standard by
RSA Security Inc. It is mostly a drop-in replacement, with the
following change:

This header file does not require any macro definitions by the user
(like CK_DEFINE_FUNCTION etc). In fact, it defines those macros
for you (if useful, some are missing, let me know if you need
more).

There is an additional API available that does comply better to the
GNU coding standard. It can be switched on by defining
CRYPTOKI_GNU before including this header file. For this, the
following changes are made to the specification:

All structure types are changed to a "struct ck_foo" where CK_FOO
is the type name in PKCS #11.

All non-structure types are changed to ck_foo_t where CK_FOO is the
lowercase version of the type name in PKCS #11. The basic types
(CK_ULONG et al.) are removed without substitute.

All members of structures are modified in the following way: Type
indication prefixes are removed, and underscore characters are
inserted before words. Then the result is lowercased.

```
   Note that function names are still in the original case, as they
   need for ABI compatibility.

   CK_FALSE, CK_TRUE and NULL_PTR are removed without substitute.  Use
   <stdbool.h>.

   If CRYPTOKI_COMPAT is defined before including this header file,
   then none of the API changes above take place, and the API is the
   one defined by the PKCS #11 standard.  */

#ifndef PKCS11_H
#define PKCS11_H 1

#if defined(__cplusplus)
extern "C" {
#endif


/* The version of cryptoki we implement.  The revision is changed with
   each modification of this file.  If you do not use the "official"
   version of this file, please consider deleting the revision macro
   (you may use a macro with a different name to keep track of your
   versions).  */
#define CRYPTOKI_VERSION_MAJOR      2
#define CRYPTOKI_VERSION_MINOR      20
#define CRYPTOKI_VERSION_REVISION   6


/* Compatibility interface is default, unless CRYPTOKI_GNU is
   given.  */
#ifndef CRYPTOKI_GNU
#ifndef CRYPTOKI_COMPAT
#define CRYPTOKI_COMPAT 1
#endif
#endif
```

```
/* System dependencies.  */

#if defined(_WIN32) || defined(CRYPTOKI_FORCE_WIN32)

/* There is a matching pop below.  */
#pragma pack(push, cryptoki, 1)

#ifdef CRYPTOKI_EXPORTS
#define CK_SPEC __declspec(dllexport)
#else
#define CK_SPEC __declspec(dllimport)
#endif

#else

#define CK_SPEC

#endif

#ifdef CRYPTOKI_COMPAT
  /* If we are in compatibility mode, switch all exposed names to the
     PKCS #11 variant.  There are corresponding #undefs below.  */

#define ck_flags_t CK_FLAGS
#define ck_version _CK_VERSION

#define ck_info _CK_INFO
#define cryptoki_version cryptokiVersion
#define manufacturer_id manufacturerID
#define library_description libraryDescription
#define library_version libraryVersion

#define ck_notification_t CK_NOTIFICATION
#define ck_slot_id_t CK_SLOT_ID

#define ck_slot_info _CK_SLOT_INFO
```

```
#define slot_description slotDescription
#define hardware_version hardwareVersion
#define firmware_version firmwareVersion

#define ck_token_info _CK_TOKEN_INFO
#define serial_number serialNumber
#define max_session_count ulMaxSessionCount
#define session_count ulSessionCount
#define max_rw_session_count ulMaxRwSessionCount
#define rw_session_count ulRwSessionCount
#define max_pin_len ulMaxPinLen
#define min_pin_len ulMinPinLen
#define total_public_memory ulTotalPublicMemory
#define free_public_memory ulFreePublicMemory
#define total_private_memory ulTotalPrivateMemory
#define free_private_memory ulFreePrivateMemory
#define utc_time utcTime

#define ck_session_handle_t CK_SESSION_HANDLE
#define ck_user_type_t CK_USER_TYPE
#define ck_state_t CK_STATE

#define ck_session_info _CK_SESSION_INFO
#define slot_id slotID
#define device_error ulDeviceError

#define ck_object_handle_t CK_OBJECT_HANDLE
#define ck_object_class_t CK_OBJECT_CLASS
#define ck_hw_feature_type_t CK_HW_FEATURE_TYPE
#define ck_key_type_t CK_KEY_TYPE
#define ck_certificate_type_t CK_CERTIFICATE_TYPE
#define ck_attribute_type_t CK_ATTRIBUTE_TYPE

#define ck_attribute _CK_ATTRIBUTE
#define value pValue
#define value_len ulValueLen
```

```
#define ck_date _CK_DATE

#define ck_mechanism_type_t CK_MECHANISM_TYPE

#define ck_rsa_pkcs_mgf_type_t CK_RSA_PKCS_MGF_TYPE

#define ck_mechanism _CK_MECHANISM
#define parameter pParameter
#define parameter_len ulParameterLen

#define ck_mechanism_info _CK_MECHANISM_INFO
#define min_key_size ulMinKeySize
#define max_key_size ulMaxKeySize

#define ck_rv_t CK_RV
#define ck_notify_t CK_NOTIFY

#define ck_function_list _CK_FUNCTION_LIST

#define ck_createmutex_t CK_CREATEMUTEX
#define ck_destroymutex_t CK_DESTROYMUTEX
#define ck_lockmutex_t CK_LOCKMUTEX
#define ck_unlockmutex_t CK_UNLOCKMUTEX

#define ck_c_initialize_args _CK_C_INITIALIZE_ARGS
#define create_mutex CreateMutex
#define destroy_mutex DestroyMutex
#define lock_mutex LockMutex
#define unlock_mutex UnlockMutex
#define reserved pReserved

#endif  /* CRYPTOKI_COMPAT */


typedef unsigned long ck_flags_t;
```

```
struct ck_version
{
  unsigned char major;
  unsigned char minor;
};


struct ck_info
{
  struct ck_version cryptoki_version;
  unsigned char manufacturer_id[32];
  ck_flags_t flags;
  unsigned char library_description[32];
  struct ck_version library_version;
};


typedef unsigned long ck_notification_t;

#define CKN_SURRENDER  (0UL)


typedef unsigned long ck_slot_id_t;


struct ck_slot_info
{
  unsigned char slot_description[64];
  unsigned char manufacturer_id[32];
  ck_flags_t flags;
  struct ck_version hardware_version;
  struct ck_version firmware_version;
};
```

```
#define CKF_TOKEN_PRESENT   (1UL << 0)
#define CKF_REMOVABLE_DEVICE   (1UL << 1)
#define CKF_HW_SLOT      (1UL << 2)
#define CKF_ARRAY_ATTRIBUTE   (1UL << 30)


struct ck_token_info
{
  unsigned char label[32];
  unsigned char manufacturer_id[32];
  unsigned char model[16];
  unsigned char serial_number[16];
  ck_flags_t flags;
  unsigned long max_session_count;
  unsigned long session_count;
  unsigned long max_rw_session_count;
  unsigned long rw_session_count;
  unsigned long max_pin_len;
  unsigned long min_pin_len;
  unsigned long total_public_memory;
  unsigned long free_public_memory;
  unsigned long total_private_memory;
  unsigned long free_private_memory;
  struct ck_version hardware_version;
  struct ck_version firmware_version;
  unsigned char utc_time[16];
};


#define CKF_RNG                (1UL << 0)
#define CKF_WRITE_PROTECTED        (1UL << 1)
#define CKF_LOGIN_REQUIRED         (1UL << 2)
#define CKF_USER_PIN_INITIALIZED      (1UL << 3)
#define CKF_RESTORE_KEY_NOT_NEEDED      (1UL << 5)
#define CKF_CLOCK_ON_TOKEN         (1UL << 6)
#define CKF_PROTECTED_AUTHENTICATION_PATH   (1UL << 8)
```

```
#define CKF_DUAL_CRYPTO_OPERATIONS      (1UL << 9)
#define CKF_TOKEN_INITIALIZED           (1UL << 10)
#define CKF_SECONDARY_AUTHENTICATION    (1UL << 11)
#define CKF_USER_PIN_COUNT_LOW          (1UL << 16)
#define CKF_USER_PIN_FINAL_TRY          (1UL << 17)
#define CKF_USER_PIN_LOCKED             (1UL << 18)
#define CKF_USER_PIN_TO_BE_CHANGED      (1UL << 19)
#define CKF_SO_PIN_COUNT_LOW            (1UL << 20)
#define CKF_SO_PIN_FINAL_TRY            (1UL << 21)
#define CKF_SO_PIN_LOCKED               (1UL << 22)
#define CKF_SO_PIN_TO_BE_CHANGED        (1UL << 23)

#define CK_UNAVAILABLE_INFORMATION  ((unsigned long) -1)
#define CK_EFFECTIVELY_INFINITE     (0UL)


typedef unsigned long ck_session_handle_t;

#define CK_INVALID_HANDLE  (0UL)


typedef unsigned long ck_user_type_t;

#define CKU_SO          (0UL)
#define CKU_USER        (1UL)
#define CKU_CONTEXT_SPECIFIC  (2UL)


typedef unsigned long ck_state_t;

#define CKS_RO_PUBLIC_SESSION  (0UL)
#define CKS_RO_USER_FUNCTIONS  (1UL)
#define CKS_RW_PUBLIC_SESSION  (2UL)
#define CKS_RW_USER_FUNCTIONS  (3UL)
#define CKS_RW_SO_FUNCTIONS  (4UL)
```

```
struct ck_session_info
{
  ck_slot_id_t slot_id;
  ck_state_t state;
  ck_flags_t flags;
  unsigned long device_error;
};

#define CKF_RW_SESSION     (1UL << 1)
#define CKF_SERIAL_SESSION  (1UL << 2)


typedef unsigned long ck_object_handle_t;


typedef unsigned long ck_object_class_t;

#define CKO_DATA     (0UL)
#define CKO_CERTIFICATE     (1UL)
#define CKO_PUBLIC_KEY     (2UL)
#define CKO_PRIVATE_KEY     (3UL)
#define CKO_SECRET_KEY     (4UL)
#define CKO_HW_FEATURE     (5UL)
#define CKO_DOMAIN_PARAMETERS  (6UL)
#define CKO_MECHANISM     (7UL)
#define CKO_VENDOR_DEFINED  (1UL << 31)


typedef unsigned long ck_hw_feature_type_t;

#define CKH_MONOTONIC_COUNTER  (1UL)
#define CKH_CLOCK     (2UL)
#define CKH_USER_INTERFACE  (3UL)
#define CKH_VENDOR_DEFINED  (1UL << 31)
```

```
typedef unsigned long ck_key_type_t;

#define CKK_RSA          (0UL)
#define CKK_DSA          (1UL)
#define CKK_DH           (2UL)
#define CKK_ECDSA        (3UL)
#define CKK_EC           (3UL)
#define CKK_X9_42_DH     (4UL)
#define CKK_KEA          (5UL)
#define CKK_GENERIC_SECRET   (0x10UL)
#define CKK_RC2          (0x11UL)
#define CKK_RC4          (0x12UL)
#define CKK_DES          (0x13UL)
#define CKK_DES2         (0x14UL)
#define CKK_DES3         (0x15UL)
#define CKK_CAST         (0x16UL)
#define CKK_CAST3        (0x17UL)
#define CKK_CAST128      (0x18UL)
#define CKK_RC5          (0x19UL)
#define CKK_IDEA         (0x1aUL)
#define CKK_SKIPJACK     (0x1bUL)
#define CKK_BATON        (0x1cUL)
#define CKK_JUNIPER      (0x1dUL)
#define CKK_CDMF         (0x1eUL)
#define CKK_AES          (0x1fUL)
#define CKK_BLOWFISH     (0x20UL)
#define CKK_TWOFISH      (0x21UL)
#define CKK_GOSTR3410    (0x30UL)
#define CKK_GOSTR3411    (0x31UL)
#define CKK_GOST28147    (0x32UL)
#define CKK_VENDOR_DEFINED   (1UL << 31)

// A mask for new GOST algorithms.
// For details visit https://tc26.ru/standarts/perevody/guidelines-the-pkcs-11-extension
s-for-implementing-the-gost-r-34-10-2012-and-gost-r-34-11-2012-russian-standards-.ht
```

```
ml
#define NSSCK_VENDOR_PKCS11_RU_TEAM      (CKK_VENDOR_DEFINED | 0
x54321000)
#define CK_VENDOR_PKCS11_RU_TEAM_TK26    NSSCK_VENDOR_PKCS11_RU_
TEAM

#define CKK_GOSTR3410_512  (CK_VENDOR_PKCS11_RU_TEAM_TK26 | 0x003)

typedef unsigned long ck_certificate_type_t;

#define CKC_X_509      (0UL)
#define CKC_X_509_ATTR_CERT  (1UL)
#define CKC_WTLS      (2UL)
#define CKC_VENDOR_DEFINED  (1UL << 31)


typedef unsigned long ck_attribute_type_t;

#define CKA_CLASS        (0UL)
#define CKA_TOKEN        (1UL)
#define CKA_PRIVATE        (2UL)
#define CKA_LABEL        (3UL)
#define CKA_APPLICATION        (0x10UL)
#define CKA_VALUE        (0x11UL)
#define CKA_OBJECT_ID        (0x12UL)
#define CKA_CERTIFICATE_TYPE      (0x80UL)
#define CKA_ISSUER        (0x81UL)
#define CKA_SERIAL_NUMBER      (0x82UL)
#define CKA_AC_ISSUER        (0x83UL)
#define CKA_OWNER        (0x84UL)
#define CKA_ATTR_TYPES        (0x85UL)
#define CKA_TRUSTED        (0x86UL)
#define CKA_CERTIFICATE_CATEGORY  (0x87UL)
#define CKA_JAVA_MIDP_SECURITY_DOMAIN  (0x88UL)
#define CKA_URL          (0x89UL)
#define CKA_HASH_OF_SUBJECT_PUBLIC_KEY  (0x8aUL)
```

```
#define CKA_HASH_OF_ISSUER_PUBLIC_KEY   (0x8bUL)
#define CKA_CHECK_VALUE         (0x90UL)
#define CKA_KEY_TYPE         (0x100UL)
#define CKA_SUBJECT         (0x101UL)
#define CKA_ID            (0x102UL)
#define CKA_SENSITIVE         (0x103UL)
#define CKA_ENCRYPT         (0x104UL)
#define CKA_DECRYPT         (0x105UL)
#define CKA_WRAP         (0x106UL)
#define CKA_UNWRAP         (0x107UL)
#define CKA_SIGN         (0x108UL)
#define CKA_SIGN_RECOVER      (0x109UL)
#define CKA_VERIFY         (0x10aUL)
#define CKA_VERIFY_RECOVER      (0x10bUL)
#define CKA_DERIVE         (0x10cUL)
#define CKA_START_DATE         (0x110UL)
#define CKA_END_DATE         (0x111UL)
#define CKA_MODULUS         (0x120UL)
#define CKA_MODULUS_BITS      (0x121UL)
#define CKA_PUBLIC_EXPONENT      (0x122UL)
#define CKA_PRIVATE_EXPONENT      (0x123UL)
#define CKA_PRIME_1         (0x124UL)
#define CKA_PRIME_2         (0x125UL)
#define CKA_EXPONENT_1         (0x126UL)
#define CKA_EXPONENT_2         (0x127UL)
#define CKA_COEFFICIENT         (0x128UL)
#define CKA_PRIME         (0x130UL)
#define CKA_SUBPRIME         (0x131UL)
#define CKA_BASE         (0x132UL)
#define CKA_PRIME_BITS         (0x133UL)
#define CKA_SUB_PRIME_BITS      (0x134UL)
#define CKA_VALUE_BITS         (0x160UL)
#define CKA_VALUE_LEN         (0x161UL)
#define CKA_EXTRACTABLE         (0x162UL)
#define CKA_LOCAL         (0x163UL)
#define CKA_NEVER_EXTRACTABLE      (0x164UL)
```

```
#define CKA_ALWAYS_SENSITIVE      (0x165UL)
#define CKA_KEY_GEN_MECHANISM     (0x166UL)
#define CKA_MODIFIABLE            (0x170UL)
#define CKA_ECDSA_PARAMS          (0x180UL)
#define CKA_EC_PARAMS             (0x180UL)
#define CKA_EC_POINT              (0x181UL)
#define CKA_SECONDARY_AUTH        (0x200UL)
#define CKA_AUTH_PIN_FLAGS        (0x201UL)
#define CKA_ALWAYS_AUTHENTICATE   (0x202UL)
#define CKA_WRAP_WITH_TRUSTED     (0x210UL)
#define CKA_GOSTR3410_PARAMS      (0x250UL)
#define CKA_GOSTR3411_PARAMS      (0x251UL)
#define CKA_GOST28147_PARAMS      (0x252UL)
#define CKA_HW_FEATURE_TYPE       (0x300UL)
#define CKA_RESET_ON_INIT         (0x301UL)
#define CKA_HAS_RESET             (0x302UL)
#define CKA_PIXEL_X               (0x400UL)
#define CKA_PIXEL_Y               (0x401UL)
#define CKA_RESOLUTION            (0x402UL)
#define CKA_CHAR_ROWS             (0x403UL)
#define CKA_CHAR_COLUMNS          (0x404UL)
#define CKA_COLOR                 (0x405UL)
#define CKA_BITS_PER_PIXEL        (0x406UL)
#define CKA_CHAR_SETS             (0x480UL)
#define CKA_ENCODING_METHODS      (0x481UL)
#define CKA_MIME_TYPES            (0x482UL)
#define CKA_MECHANISM_TYPE        (0x500UL)
#define CKA_REQUIRED_CMS_ATTRIBUTES   (0x501UL)
#define CKA_DEFAULT_CMS_ATTRIBUTES    (0x502UL)
#define CKA_SUPPORTED_CMS_ATTRIBUTES  (0x503UL)
#define CKA_WRAP_TEMPLATE         (CKF_ARRAY_ATTRIBUTE | 0x211UL)
#define CKA_UNWRAP_TEMPLATE       (CKF_ARRAY_ATTRIBUTE | 0x212UL)
#define CKA_OTP_FORMAT            (0x220UL)
#define CKA_OTP_LENGTH            (0x221UL)
#define CKA_OTP_TIME_INTERVAL     (0x222UL)
#define CKA_OTP_USER_FRIENDLY_MODE    (0x223UL)
```

```
#define CKA_OTP_CHALLENGE_REQUIREMENT   (0x224UL)
#define CKA_OTP_TIME_REQUIREMENT   (0x225UL)
#define CKA_OTP_COUNTER_REQUIREMENT   (0x226UL)
#define CKA_OTP_PIN_REQUIREMENT      (0x227UL)
#define CKA_OTP_USER_IDENTIFIER      (0x22AUL)
#define CKA_OTP_SERVICE_IDENTIFIER   (0x22BUL)
#define CKA_OTP_SERVICE_LOGO      (0x22CUL)
#define CKA_OTP_SERVICE_LOGO_TYPE   (0x22DUL)
#define CKA_OTP_COUNTER         (0x22EUL)
#define CKA_OTP_TIME         (0x22FUL)
#define CKA_ALLOWED_MECHANISMS      (CKF_ARRAY_ATTRIBUTE | 0x600U
L)
#define CKA_VENDOR_DEFINED      (1UL << 31)


struct ck_attribute
{
  ck_attribute_type_t type;
  void *value;
  unsigned long value_len;
};


struct ck_date
{
  unsigned char year[4];
  unsigned char month[2];
  unsigned char day[2];
};


typedef unsigned long ck_mechanism_type_t;

#define CKM_RSA_PKCS_KEY_PAIR_GEN   (0UL)
#define CKM_RSA_PKCS         (1UL)
#define CKM_RSA_9796         (2UL)
```

```
#define CKM_RSA_X_509          (3UL)
#define CKM_MD2_RSA_PKCS       (4UL)
#define CKM_MD5_RSA_PKCS       (5UL)
#define CKM_SHA1_RSA_PKCS      (6UL)
#define CKM_RIPEMD128_RSA_PKCS      (7UL)
#define CKM_RIPEMD160_RSA_PKCS      (8UL)
#define CKM_RSA_PKCS_OAEP      (9UL)
#define CKM_RSA_X9_31_KEY_PAIR_GEN   (0xaUL)
#define CKM_RSA_X9_31          (0xbUL)
#define CKM_SHA1_RSA_X9_31      (0xcUL)
#define CKM_RSA_PKCS_PSS       (0xdUL)
#define CKM_SHA1_RSA_PKCS_PSS      (0xeUL)
#define CKM_DSA_KEY_PAIR_GEN      (0x10UL)
#define  CKM_DSA           (0x11UL)
#define CKM_DSA_SHA1          (0x12UL)
#define CKM_DSA_SHA224          (0x13UL)
#define CKM_DSA_SHA256          (0x14UL)
#define CKM_DSA_SHA384          (0x15UL)
#define CKM_DSA_SHA512          (0x16UL)
#define CKM_DH_PKCS_KEY_PAIR_GEN   (0x20UL)
#define CKM_DH_PKCS_DERIVE      (0x21UL)
#define  CKM_X9_42_DH_KEY_PAIR_GEN   (0x30UL)
#define CKM_X9_42_DH_DERIVE      (0x31UL)
#define CKM_X9_42_DH_HYBRID_DERIVE   (0x32UL)
#define CKM_X9_42_MQV_DERIVE      (0x33UL)
#define CKM_SHA256_RSA_PKCS      (0x40UL)
#define CKM_SHA384_RSA_PKCS      (0x41UL)
#define CKM_SHA512_RSA_PKCS      (0x42UL)
#define CKM_SHA256_RSA_PKCS_PSS      (0x43UL)
#define CKM_SHA384_RSA_PKCS_PSS      (0x44UL)
#define CKM_SHA512_RSA_PKCS_PSS      (0x45UL)
#define CKM_SHA224_RSA_PKCS      (0x46UL)
#define CKM_SHA224_RSA_PKCS_PSS      (0x47UL)
#define CKM_RC2_KEY_GEN          (0x100UL)
#define CKM_RC2_ECB          (0x101UL)
#define  CKM_RC2_CBC          (0x102UL)
```

```
#define   CKM_RC2_MAC          (0x103UL)
#define  CKM_RC2_MAC_GENERAL      (0x104UL)
#define  CKM_RC2_CBC_PAD          (0x105UL)
#define  CKM_RC4_KEY_GEN          (0x110UL)
#define  CKM_RC4             (0x111UL)
#define  CKM_DES_KEY_GEN          (0x120UL)
#define  CKM_DES_ECB          (0x121UL)
#define  CKM_DES_CBC          (0x122UL)
#define  CKM_DES_MAC          (0x123UL)
#define  CKM_DES_MAC_GENERAL      (0x124UL)
#define  CKM_DES_CBC_PAD          (0x125UL)
#define  CKM_DES2_KEY_GEN      (0x130UL)
#define  CKM_DES3_KEY_GEN      (0x131UL)
#define  CKM_DES3_ECB          (0x132UL)
#define  CKM_DES3_CBC          (0x133UL)
#define  CKM_DES3_MAC          (0x134UL)
#define  CKM_DES3_MAC_GENERAL      (0x135UL)
#define  CKM_DES3_CBC_PAD      (0x136UL)
#define  CKM_DES3_CMAC          (0x138UL)
#define  CKM_CDMF_KEY_GEN      (0x140UL)
#define  CKM_CDMF_ECB          (0x141UL)
#define  CKM_CDMF_CBC          (0x142UL)
#define  CKM_CDMF_MAC          (0x143UL)
#define  CKM_CDMF_MAC_GENERAL      (0x144UL)
#define  CKM_CDMF_CBC_PAD      (0x145UL)
#define  CKM_MD2             (0x200UL)
#define  CKM_MD2_HMAC          (0x201UL)
#define  CKM_MD2_HMAC_GENERAL      (0x202UL)
#define  CKM_MD5             (0x210UL)
#define  CKM_MD5_HMAC          (0x211UL)
#define  CKM_MD5_HMAC_GENERAL      (0x212UL)
#define  CKM_SHA_1          (0x220UL)
#define  CKM_SHA_1_HMAC          (0x221UL)
#define  CKM_SHA_1_HMAC_GENERAL      (0x222UL)
#define  CKM_RIPEMD128          (0x230UL)
#define  CKM_RIPEMD128_HMAC      (0x231UL)
```

```
#define CKM_RIPEMD128_HMAC_GENERAL  (0x232UL)
#define CKM_RIPEMD160       (0x240UL)
#define CKM_RIPEMD160_HMAC     (0x241UL)
#define CKM_RIPEMD160_HMAC_GENERAL  (0x242UL)
#define CKM_SHA256       (0x250UL)
#define CKM_SHA256_HMAC       (0x251UL)
#define CKM_SHA256_HMAC_GENERAL     (0x252UL)
#define CKM_SHA224       (0x255UL)
#define CKM_SHA224_HMAC       (0x256UL)
#define CKM_SHA224_HMAC_GENERAL     (0x257UL)
#define CKM_SHA384       (0x260UL)
#define CKM_SHA384_HMAC       (0x261UL)
#define CKM_SHA384_HMAC_GENERAL     (0x262UL)
#define CKM_SHA512       (0x270UL)
#define CKM_SHA512_HMAC       (0x271UL)
#define CKM_SHA512_HMAC_GENERAL     (0x272UL)
#define CKM_CAST_KEY_GEN     (0x300UL)
#define CKM_CAST_ECB       (0x301UL)
#define CKM_CAST_CBC       (0x302UL)
#define CKM_CAST_MAC       (0x303UL)
#define CKM_CAST_MAC_GENERAL     (0x304UL)
#define CKM_CAST_CBC_PAD     (0x305UL)
#define CKM_CAST3_KEY_GEN     (0x310UL)
#define CKM_CAST3_ECB       (0x311UL)
#define CKM_CAST3_CBC       (0x312UL)
#define CKM_CAST3_MAC       (0x313UL)
#define CKM_CAST3_MAC_GENERAL     (0x314UL)
#define CKM_CAST3_CBC_PAD     (0x315UL)
#define CKM_CAST5_KEY_GEN     (0x320UL)
#define CKM_CAST128_KEY_GEN     (0x320UL)
#define CKM_CAST5_ECB       (0x321UL)
#define CKM_CAST128_ECB       (0x321UL)
#define CKM_CAST5_CBC       (0x322UL)
#define CKM_CAST128_CBC       (0x322UL)
#define CKM_CAST5_MAC       (0x323UL)
#define   CKM_CAST128_MAC       (0x323UL)
```

```
#define CKM_CAST5_MAC_GENERAL      (0x324UL)
#define CKM_CAST128_MAC_GENERAL      (0x324UL)
#define CKM_CAST5_CBC_PAD      (0x325UL)
#define CKM_CAST128_CBC_PAD      (0x325UL)
#define CKM_RC5_KEY_GEN      (0x330UL)
#define CKM_RC5_ECB      (0x331UL)
#define CKM_RC5_CBC      (0x332UL)
#define CKM_RC5_MAC      (0x333UL)
#define CKM_RC5_MAC_GENERAL      (0x334UL)
#define CKM_RC5_CBC_PAD      (0x335UL)
#define CKM_IDEA_KEY_GEN      (0x340UL)
#define CKM_IDEA_ECB      (0x341UL)
#define  CKM_IDEA_CBC      (0x342UL)
#define CKM_IDEA_MAC      (0x343UL)
#define CKM_IDEA_MAC_GENERAL      (0x344UL)
#define CKM_IDEA_CBC_PAD      (0x345UL)
#define CKM_GENERIC_SECRET_KEY_GEN   (0x350UL)
#define CKM_CONCATENATE_BASE_AND_KEY   (0x360UL)
#define CKM_CONCATENATE_BASE_AND_DATA   (0x362UL)
#define CKM_CONCATENATE_DATA_AND_BASE   (0x363UL)
#define CKM_XOR_BASE_AND_DATA      (0x364UL)
#define CKM_EXTRACT_KEY_FROM_KEY   (0x365UL)
#define CKM_SSL3_PRE_MASTER_KEY_GEN   (0x370UL)
#define CKM_SSL3_MASTER_KEY_DERIVE   (0x371UL)
#define CKM_SSL3_KEY_AND_MAC_DERIVE   (0x372UL)
#define CKM_SSL3_MASTER_KEY_DERIVE_DH   (0x373UL)
#define CKM_TLS_PRE_MASTER_KEY_GEN   (0x374UL)
#define CKM_TLS_MASTER_KEY_DERIVE   (0x375UL)
#define CKM_TLS_KEY_AND_MAC_DERIVE   (0x376UL)
#define CKM_TLS_MASTER_KEY_DERIVE_DH   (0x377UL)
#define CKM_SSL3_MD5_MAC      (0x380UL)
#define CKM_SSL3_SHA1_MAC      (0x381UL)
#define CKM_MD5_KEY_DERIVATION      (0x390UL)
#define CKM_MD2_KEY_DERIVATION      (0x391UL)
#define CKM_SHA1_KEY_DERIVATION      (0x392UL)
#define CKM_PBE_MD2_DES_CBC      (0x3a0UL)
```

```
#define CKM_PBE_MD5_DES_CBC      (0x3a1UL)
#define CKM_PBE_MD5_CAST_CBC     (0x3a2UL)
#define CKM_PBE_MD5_CAST3_CBC    (0x3a3UL)
#define CKM_PBE_MD5_CAST5_CBC    (0x3a4UL)
#define CKM_PBE_MD5_CAST128_CBC     (0x3a4UL)
#define CKM_PBE_SHA1_CAST5_CBC    (0x3a5UL)
#define CKM_PBE_SHA1_CAST128_CBC  (0x3a5UL)
#define CKM_PBE_SHA1_RC4_128     (0x3a6UL)
#define CKM_PBE_SHA1_RC4_40      (0x3a7UL)
#define CKM_PBE_SHA1_DES3_EDE_CBC  (0x3a8UL)
#define CKM_PBE_SHA1_DES2_EDE_CBC  (0x3a9UL)
#define CKM_PBE_SHA1_RC2_128_CBC  (0x3aaUL)
#define CKM_PBE_SHA1_RC2_40_CBC    (0x3abUL)
#define CKM_PKCS5_PBKD2        (0x3b0UL)
#define CKM_PBA_SHA1_WITH_SHA1_HMAC  (0x3c0UL)
#define CKM_KEY_WRAP_LYNKS     (0x400UL)
#define CKM_KEY_WRAP_SET_OAEP     (0x401UL)
#define CKM_SKIPJACK_KEY_GEN     (0x1000UL)
#define CKM_SKIPJACK_ECB64     (0x1001UL)
#define CKM_SKIPJACK_CBC64     (0x1002UL)
#define CKM_SKIPJACK_OFB64     (0x1003UL)
#define CKM_SKIPJACK_CFB64     (0x1004UL)
#define CKM_SKIPJACK_CFB32     (0x1005UL)
#define CKM_SKIPJACK_CFB16     (0x1006UL)
#define CKM_SKIPJACK_CFB8      (0x1007UL)
#define CKM_SKIPJACK_WRAP      (0x1008UL)
#define CKM_SKIPJACK_PRIVATE_WRAP  (0x1009UL)
#define CKM_SKIPJACK_RELAYX      (0x100aUL)
#define CKM_KEA_KEY_PAIR_GEN     (0x1010UL)
#define CKM_KEA_KEY_DERIVE     (0x1011UL)
#define CKM_FORTEZZA_TIMESTAMP     (0x1020UL)
#define CKM_BATON_KEY_GEN     (0x1030UL)
#define CKM_BATON_ECB128      (0x1031UL)
#define CKM_BATON_ECB96        (0x1032UL)
#define CKM_BATON_CBC128      (0x1033UL)
#define CKM_BATON_COUNTER      (0x1034UL)
```

```
#define CKM_BATON_SHUFFLE      (0x1035UL)
#define CKM_BATON_WRAP         (0x1036UL)
#define CKM_ECDSA_KEY_PAIR_GEN      (0x1040UL)
#define CKM_EC_KEY_PAIR_GEN      (0x1040UL)
#define CKM_ECDSA         (0x1041UL)
#define CKM_ECDSA_SHA1         (0x1042UL)
#define CKM_ECDSA_SHA224      (0x1043UL)
#define CKM_ECDSA_SHA256      (0x1044UL)
#define CKM_ECDSA_SHA384      (0x1045UL)
#define CKM_ECDSA_SHA512      (0x1046UL)
#define CKM_ECDH1_DERIVE      (0x1050UL)
#define CKM_ECDH1_COFACTOR_DERIVE   (0x1051UL)
#define CKM_ECMQV_DERIVE      (0x1052UL)
#define CKM_JUNIPER_KEY_GEN      (0x1060UL)
#define CKM_JUNIPER_ECB128      (0x1061UL)
#define CKM_JUNIPER_CBC128      (0x1062UL)
#define CKM_JUNIPER_COUNTER      (0x1063UL)
#define CKM_JUNIPER_SHUFFLE      (0x1064UL)
#define CKM_JUNIPER_WRAP      (0x1065UL)
#define CKM_FASTHASH         (0x1070UL)
#define CKM_AES_KEY_GEN         (0x1080UL)
#define CKM_AES_ECB         (0x1081UL)
#define CKM_AES_CBC         (0x1082UL)
#define CKM_AES_MAC         (0x1083UL)
#define CKM_AES_MAC_GENERAL      (0x1084UL)
#define CKM_AES_CBC_PAD         (0x1085UL)
#define CKM_AES_CTR            (0x1086UL)
#define CKM_AES_GCM            (0x1087UL)
#define CKM_AES_CCM            (0x1088UL)
#define CKM_AES_CTS            (0x1089UL)
#define CKM_AES_CMAC         (0x108AUL)
#define CKM_BLOWFISH_KEY_GEN      (0x1090UL)
#define CKM_BLOWFISH_CBC            (0x1091UL)
#define CKM_TWOFISH_KEY_GEN       (0x1092UL)
#define CKM_TWOFISH_CBC            (0x1093UL)
#define CKM_DES_ECB_ENCRYPT_DATA   (0x1100UL)
```

```
#define CKM_DES_CBC_ENCRYPT_DATA    (0x1101UL)
#define CKM_DES3_ECB_ENCRYPT_DATA   (0x1102UL)
#define CKM_DES3_CBC_ENCRYPT_DATA   (0x1103UL)
#define CKM_AES_ECB_ENCRYPT_DATA    (0x1104UL)
#define CKM_AES_CBC_ENCRYPT_DATA    (0x1105UL)
#define CKM_GOSTR3410_KEY_PAIR_GEN  (0x1200UL)
#define CKM_GOSTR3410               (0x1201UL)
#define CKM_GOSTR3410_WITH_GOSTR3411  (0x1202UL)
#define CKM_GOSTR3410_KEY_WRAP      (0x1203UL)
#define CKM_GOSTR3410_DERIVE        (0x1204UL)
#define CKM_GOSTR3410_512_KEY_PAIR_GEN  (CK_VENDOR_PKCS11_RU_TEA
M_TK26 | 0x005)
#define CKM_GOSTR3410_512  (CK_VENDOR_PKCS11_RU_TEAM_TK26 | 0x006)
#define CKM_GOSTR3410_12_DERIVE  (CK_VENDOR_PKCS11_RU_TEAM_TK26
| 0x007)
#define CKM_GOSTR3410_WITH_GOSTR3411_12_256  (CK_VENDOR_PKCS11_RU
_TEAM_TK26 | 0x008)
#define CKM_GOSTR3410_WITH_GOSTR3411_12_512  (CK_VENDOR_PKCS11_RU
_TEAM_TK26 | 0x009)
#define CKM_GOSTR3411               (0x1210UL)
#define CKM_GOSTR3411_HMAC          (0x1211UL)
#define CKM_GOSTR3411_12_256  (CK_VENDOR_PKCS11_RU_TEAM_TK26 | 0x0
12)
#define CKM_GOSTR3411_12_512  (CK_VENDOR_PKCS11_RU_TEAM_TK26 | 0x0
13)
#define CKM_GOSTR3411_12_256_HMAC  (CK_VENDOR_PKCS11_RU_TEAM_TK2
6 | 0x014)
#define CKM_GOSTR3411_12_512_HMAC  (CK_VENDOR_PKCS11_RU_TEAM_TK2
6 | 0x015)
#define CKM_GOST28147_KEY_GEN    (0x1220UL)
#define CKM_GOST28147_ECB           (0x1221UL)
#define CKM_GOST28147               (0x1222UL)
#define CKM_GOST28147_MAC           (0x1223UL)
#define CKM_GOST28147_KEY_WRAP   (0x1224UL)

#define CKM_DSA_PARAMETER_GEN       (0x2000UL)
```

```
#define CKM_DH_PKCS_PARAMETER_GEN  (0x2001UL)
#define CKM_X9_42_DH_PARAMETER_GEN  (0x2002UL)
#define CKM_AES_KEY_WRAP     (0x2109UL)
#define CKM_VENDOR_DEFINED     (1UL << 31)


struct ck_mechanism
{
  ck_mechanism_type_t mechanism;
  void *parameter;
  unsigned long parameter_len;
};


struct ck_mechanism_info
{
  unsigned long min_key_size;
  unsigned long max_key_size;
  ck_flags_t flags;
};

#define CKF_HW        (1UL << 0)
#define CKF_ENCRYPT     (1UL << 8)
#define CKF_DECRYPT     (1UL << 9)
#define CKF_DIGEST     (1UL << 10)
#define CKF_SIGN     (1UL << 11)
#define CKF_SIGN_RECOVER  (1UL << 12)
#define CKF_VERIFY     (1UL << 13)
#define CKF_VERIFY_RECOVER  (1UL << 14)
#define CKF_GENERATE     (1UL << 15)
#define CKF_GENERATE_KEY_PAIR  (1UL << 16)
#define CKF_WRAP     (1UL << 17)
#define CKF_UNWRAP     (1UL << 18)
#define CKF_DERIVE     (1UL << 19)
#define CKF_EXTENSION     (1UL << 31)
```

```
#define CKF_EC_F_P          (1UL << 20)
#define CKF_EC_F_2M         (1UL << 21)
#define CKF_EC_ECPARAMETERS  (1UL << 22)
#define CKF_EC_NAMEDCURVE   (1UL << 23)
#define CKF_EC_UNCOMPRESS   (1UL << 24)
#define CKF_EC_COMPRESS     (1UL << 25)

/* Flags for C_WaitForSlotEvent.  */
#define CKF_DONT_BLOCK          (1UL)

/* Flags for Key derivation */
#define CKD_NULL        (0x1UL)
#define CKD_SHA1_KDF        (0x2UL)
#define CKD_SHA224_KDF      (0x5UL)
#define CKD_SHA256_KDF      (0x6UL)
#define CKD_SHA384_KDF      (0x7UL)
#define CKD_SHA512_KDF      (0x8UL)

typedef struct CK_ECDH1_DERIVE_PARAMS {
  unsigned long  kdf;
  unsigned long  ulSharedDataLen;
  unsigned char *  pSharedData;
  unsigned long  ulPublicDataLen;
  unsigned char *  pPublicData;
} CK_ECDH1_DERIVE_PARAMS;

typedef struct CK_ECMQV_DERIVE_PARAMS {
  unsigned long kdf;
  unsigned long ulSharedDataLen;
  unsigned char * pSharedData;
  unsigned long ulPublicDataLen;
  unsigned char * pPublicData;
  unsigned long ulPrivateDataLen;
  CK_OBJECT_HANDLE hPrivateData;
  unsigned long ulPublicDataLen2;
  unsigned char * pPublicData2;
```

```
  CK_OBJECT_HANDLE publicKey;
} CK_ECMQV_DERIVE_PARAMS;

typedef unsigned long ck_rsa_pkcs_mgf_type_t;
typedef unsigned long CK_RSA_PKCS_OAEP_SOURCE_TYPE;

typedef struct CK_RSA_PKCS_OAEP_PARAMS {
  CK_MECHANISM_TYPE hashAlg;
  CK_RSA_PKCS_MGF_TYPE mgf;
  CK_RSA_PKCS_OAEP_SOURCE_TYPE source;
  void *pSourceData;
  unsigned long ulSourceDataLen;
} CK_RSA_PKCS_OAEP_PARAMS;

typedef struct CK_RSA_PKCS_PSS_PARAMS {
  ck_mechanism_type_t hashAlg;
  CK_RSA_PKCS_MGF_TYPE mgf;
  unsigned long sLen;
} CK_RSA_PKCS_PSS_PARAMS;

#define CKG_MGF1_SHA1         (0x00000001UL)
#define CKG_MGF1_SHA224       (0x00000005UL)
#define CKG_MGF1_SHA256       (0x00000002UL)
#define CKG_MGF1_SHA384       (0x00000003UL)
#define CKG_MGF1_SHA512       (0x00000004UL)

#define CKZ_DATA_SPECIFIED    (0x00000001UL)

typedef struct CK_GCM_PARAMS {
  void * pIv;
  unsigned long ulIvLen;
  unsigned long ulIvBits;
  void * pAAD;
  unsigned long ulAADLen;
  unsigned long ulTagBits;
} CK_GCM_PARAMS;
```

```
typedef unsigned long ck_rv_t;


typedef ck_rv_t (*ck_notify_t) (ck_session_handle_t session,
          ck_notification_t event, void *application);

/* Forward reference.  */
struct ck_function_list;

#define _CK_DECLARE_FUNCTION(name, args)  ¥
typedef ck_rv_t (*CK_ ## name) args;     ¥
ck_rv_t CK_SPEC name args

_CK_DECLARE_FUNCTION (C_Initialize, (void *init_args));
_CK_DECLARE_FUNCTION (C_Finalize, (void *reserved));
_CK_DECLARE_FUNCTION (C_GetInfo, (struct ck_info *info));
_CK_DECLARE_FUNCTION (C_GetFunctionList,
          (struct ck_function_list **function_list));

_CK_DECLARE_FUNCTION (C_GetSlotList,
          (unsigned char token_present, ck_slot_id_t *slot_list,
           unsigned long *count));
_CK_DECLARE_FUNCTION (C_GetSlotInfo,
          (ck_slot_id_t slot_id, struct ck_slot_info *info));
_CK_DECLARE_FUNCTION (C_GetTokenInfo,
          (ck_slot_id_t slot_id, struct ck_token_info *info));
_CK_DECLARE_FUNCTION (C_WaitForSlotEvent,
          (ck_flags_t flags, ck_slot_id_t *slot, void *reserved));
_CK_DECLARE_FUNCTION (C_GetMechanismList,
          (ck_slot_id_t slot_id,
           ck_mechanism_type_t *mechanism_list,
           unsigned long *count));
_CK_DECLARE_FUNCTION (C_GetMechanismInfo,
          (ck_slot_id_t slot_id, ck_mechanism_type_t type,
           struct ck_mechanism_info *info));
```

```
_CK_DECLARE_FUNCTION (C_InitToken,
            (ck_slot_id_t slot_id, unsigned char *pin,
             unsigned long pin_len, unsigned char *label));
_CK_DECLARE_FUNCTION (C_InitPIN,
            (ck_session_handle_t session, unsigned char *pin,
             unsigned long pin_len));
_CK_DECLARE_FUNCTION (C_SetPIN,
            (ck_session_handle_t session, unsigned char *old_pin,
             unsigned long old_len, unsigned char *new_pin,
             unsigned long new_len));

_CK_DECLARE_FUNCTION (C_OpenSession,
            (ck_slot_id_t slot_id, ck_flags_t flags,
             void *application, ck_notify_t notify,
             ck_session_handle_t *session));
_CK_DECLARE_FUNCTION (C_CloseSession, (ck_session_handle_t session));
_CK_DECLARE_FUNCTION (C_CloseAllSessions, (ck_slot_id_t slot_id));
_CK_DECLARE_FUNCTION (C_GetSessionInfo,
            (ck_session_handle_t session,
             struct ck_session_info *info));
_CK_DECLARE_FUNCTION (C_GetOperationState,
            (ck_session_handle_t session,
             unsigned char *operation_state,
             unsigned long *operation_state_len));
_CK_DECLARE_FUNCTION (C_SetOperationState,
            (ck_session_handle_t session,
             unsigned char *operation_state,
             unsigned long operation_state_len,
             ck_object_handle_t encryption_key,
             ck_object_handle_t authentication_key));
_CK_DECLARE_FUNCTION (C_Login,
            (ck_session_handle_t session, ck_user_type_t user_type,
             unsigned char *pin, unsigned long pin_len));
_CK_DECLARE_FUNCTION (C_Logout, (ck_session_handle_t session));

_CK_DECLARE_FUNCTION (C_CreateObject,
```

```
        (ck_session_handle_t session,
         struct ck_attribute *templ,
         unsigned long count, ck_object_handle_t *object));
_CK_DECLARE_FUNCTION (C_CopyObject,
        (ck_session_handle_t session, ck_object_handle_t object,
         struct ck_attribute *templ, unsigned long count,
         ck_object_handle_t *new_object));
_CK_DECLARE_FUNCTION (C_DestroyObject,
        (ck_session_handle_t session,
         ck_object_handle_t object));
_CK_DECLARE_FUNCTION (C_GetObjectSize,
        (ck_session_handle_t session,
         ck_object_handle_t object,
         unsigned long *size));
_CK_DECLARE_FUNCTION (C_GetAttributeValue,
        (ck_session_handle_t session,
         ck_object_handle_t object,
         struct ck_attribute *templ,
         unsigned long count));
_CK_DECLARE_FUNCTION (C_SetAttributeValue,
        (ck_session_handle_t session,
         ck_object_handle_t object,
         struct ck_attribute *templ,
         unsigned long count));
_CK_DECLARE_FUNCTION (C_FindObjectsInit,
        (ck_session_handle_t session,
         struct ck_attribute *templ,
         unsigned long count));
_CK_DECLARE_FUNCTION (C_FindObjects,
        (ck_session_handle_t session,
         ck_object_handle_t *object,
         unsigned long max_object_count,
         unsigned long *object_count));
_CK_DECLARE_FUNCTION (C_FindObjectsFinal,
        (ck_session_handle_t session));
```

```
_CK_DECLARE_FUNCTION (C_EncryptInit,
            (ck_session_handle_t session,
             struct ck_mechanism *mechanism,
             ck_object_handle_t key));
_CK_DECLARE_FUNCTION (C_Encrypt,
            (ck_session_handle_t session,
             unsigned char *data, unsigned long data_len,
             unsigned char *encrypted_data,
             unsigned long *encrypted_data_len));
_CK_DECLARE_FUNCTION (C_EncryptUpdate,
            (ck_session_handle_t session,
             unsigned char *part, unsigned long part_len,
             unsigned char *encrypted_part,
             unsigned long *encrypted_part_len));
_CK_DECLARE_FUNCTION (C_EncryptFinal,
            (ck_session_handle_t session,
             unsigned char *last_encrypted_part,
             unsigned long *last_encrypted_part_len));

_CK_DECLARE_FUNCTION (C_DecryptInit,
            (ck_session_handle_t session,
             struct ck_mechanism *mechanism,
             ck_object_handle_t key));
_CK_DECLARE_FUNCTION (C_Decrypt,
            (ck_session_handle_t session,
             unsigned char *encrypted_data,
             unsigned long encrypted_data_len,
             unsigned char *data, unsigned long *data_len));
_CK_DECLARE_FUNCTION (C_DecryptUpdate,
            (ck_session_handle_t session,
             unsigned char *encrypted_part,
             unsigned long encrypted_part_len,
             unsigned char *part, unsigned long *part_len));
_CK_DECLARE_FUNCTION (C_DecryptFinal,
            (ck_session_handle_t session,
             unsigned char *last_part,
```

```
            unsigned long *last_part_len));

_CK_DECLARE_FUNCTION (C_DigestInit,
            (ck_session_handle_t session,
             struct ck_mechanism *mechanism));
_CK_DECLARE_FUNCTION (C_Digest,
            (ck_session_handle_t session,
             unsigned char *data, unsigned long data_len,
             unsigned char *digest,
             unsigned long *digest_len));
_CK_DECLARE_FUNCTION (C_DigestUpdate,
            (ck_session_handle_t session,
             unsigned char *part, unsigned long part_len));
_CK_DECLARE_FUNCTION (C_DigestKey,
            (ck_session_handle_t session, ck_object_handle_t key));
_CK_DECLARE_FUNCTION (C_DigestFinal,
            (ck_session_handle_t session,
             unsigned char *digest,
             unsigned long *digest_len));

_CK_DECLARE_FUNCTION (C_SignInit,
            (ck_session_handle_t session,
             struct ck_mechanism *mechanism,
             ck_object_handle_t key));
_CK_DECLARE_FUNCTION (C_Sign,
            (ck_session_handle_t session,
             unsigned char *data, unsigned long data_len,
             unsigned char *signature,
             unsigned long *signature_len));
_CK_DECLARE_FUNCTION (C_SignUpdate,
            (ck_session_handle_t session,
             unsigned char *part, unsigned long part_len));
_CK_DECLARE_FUNCTION (C_SignFinal,
            (ck_session_handle_t session,
             unsigned char *signature,
             unsigned long *signature_len));
```

```
_CK_DECLARE_FUNCTION (C_SignRecoverInit,
            (ck_session_handle_t session,
             struct ck_mechanism *mechanism,
             ck_object_handle_t key));
_CK_DECLARE_FUNCTION (C_SignRecover,
            (ck_session_handle_t session,
             unsigned char *data, unsigned long data_len,
             unsigned char *signature,
             unsigned long *signature_len));

_CK_DECLARE_FUNCTION (C_VerifyInit,
            (ck_session_handle_t session,
             struct ck_mechanism *mechanism,
             ck_object_handle_t key));
_CK_DECLARE_FUNCTION (C_Verify,
            (ck_session_handle_t session,
             unsigned char *data, unsigned long data_len,
             unsigned char *signature,
             unsigned long signature_len));
_CK_DECLARE_FUNCTION (C_VerifyUpdate,
            (ck_session_handle_t session,
             unsigned char *part, unsigned long part_len));
_CK_DECLARE_FUNCTION (C_VerifyFinal,
            (ck_session_handle_t session,
             unsigned char *signature,
             unsigned long signature_len));
_CK_DECLARE_FUNCTION (C_VerifyRecoverInit,
            (ck_session_handle_t session,
             struct ck_mechanism *mechanism,
             ck_object_handle_t key));
_CK_DECLARE_FUNCTION (C_VerifyRecover,
            (ck_session_handle_t session,
             unsigned char *signature,
             unsigned long signature_len,
             unsigned char *data,
             unsigned long *data_len));
```

```
_CK_DECLARE_FUNCTION (C_DigestEncryptUpdate,
            (ck_session_handle_t session,
             unsigned char *part, unsigned long part_len,
             unsigned char *encrypted_part,
             unsigned long *encrypted_part_len));
_CK_DECLARE_FUNCTION (C_DecryptDigestUpdate,
            (ck_session_handle_t session,
             unsigned char *encrypted_part,
             unsigned long encrypted_part_len,
             unsigned char *part,
             unsigned long *part_len));
_CK_DECLARE_FUNCTION (C_SignEncryptUpdate,
            (ck_session_handle_t session,
             unsigned char *part, unsigned long part_len,
             unsigned char *encrypted_part,
             unsigned long *encrypted_part_len));
_CK_DECLARE_FUNCTION (C_DecryptVerifyUpdate,
            (ck_session_handle_t session,
             unsigned char *encrypted_part,
             unsigned long encrypted_part_len,
             unsigned char *part,
             unsigned long *part_len));

_CK_DECLARE_FUNCTION (C_GenerateKey,
            (ck_session_handle_t session,
             struct ck_mechanism *mechanism,
             struct ck_attribute *templ,
             unsigned long count,
             ck_object_handle_t *key));
_CK_DECLARE_FUNCTION (C_GenerateKeyPair,
            (ck_session_handle_t session,
             struct ck_mechanism *mechanism,
             struct ck_attribute *public_key_template,
             unsigned long public_key_attribute_count,
             struct ck_attribute *private_key_template,
```

```
            unsigned long private_key_attribute_count,
            ck_object_handle_t *public_key,
            ck_object_handle_t *private_key));
_CK_DECLARE_FUNCTION (C_WrapKey,
           (ck_session_handle_t session,
            struct ck_mechanism *mechanism,
            ck_object_handle_t wrapping_key,
            ck_object_handle_t key,
            unsigned char *wrapped_key,
            unsigned long *wrapped_key_len));
_CK_DECLARE_FUNCTION (C_UnwrapKey,
           (ck_session_handle_t session,
            struct ck_mechanism *mechanism,
            ck_object_handle_t unwrapping_key,
            unsigned char *wrapped_key,
            unsigned long wrapped_key_len,
            struct ck_attribute *templ,
            unsigned long attribute_count,
            ck_object_handle_t *key));
_CK_DECLARE_FUNCTION (C_DeriveKey,
           (ck_session_handle_t session,
            struct ck_mechanism *mechanism,
            ck_object_handle_t base_key,
            struct ck_attribute *templ,
            unsigned long attribute_count,
            ck_object_handle_t *key));

_CK_DECLARE_FUNCTION (C_SeedRandom,
           (ck_session_handle_t session, unsigned char *seed,
            unsigned long seed_len));
_CK_DECLARE_FUNCTION (C_GenerateRandom,
           (ck_session_handle_t session,
            unsigned char *random_data,
            unsigned long random_len));

_CK_DECLARE_FUNCTION (C_GetFunctionStatus, (ck_session_handle_t session));
```

```
_CK_DECLARE_FUNCTION (C_CancelFunction, (ck_session_handle_t session));


struct ck_function_list
{
  struct ck_version version;
  CK_C_Initialize C_Initialize;
  CK_C_Finalize C_Finalize;
  CK_C_GetInfo C_GetInfo;
  CK_C_GetFunctionList C_GetFunctionList;
  CK_C_GetSlotList C_GetSlotList;
  CK_C_GetSlotInfo C_GetSlotInfo;
  CK_C_GetTokenInfo C_GetTokenInfo;
  CK_C_GetMechanismList C_GetMechanismList;
  CK_C_GetMechanismInfo C_GetMechanismInfo;
  CK_C_InitToken C_InitToken;
  CK_C_InitPIN C_InitPIN;
  CK_C_SetPIN C_SetPIN;
  CK_C_OpenSession C_OpenSession;
  CK_C_CloseSession C_CloseSession;
  CK_C_CloseAllSessions C_CloseAllSessions;
  CK_C_GetSessionInfo C_GetSessionInfo;
  CK_C_GetOperationState C_GetOperationState;
  CK_C_SetOperationState C_SetOperationState;
  CK_C_Login C_Login;
  CK_C_Logout C_Logout;
  CK_C_CreateObject C_CreateObject;
  CK_C_CopyObject C_CopyObject;
  CK_C_DestroyObject C_DestroyObject;
  CK_C_GetObjectSize C_GetObjectSize;
  CK_C_GetAttributeValue C_GetAttributeValue;
  CK_C_SetAttributeValue C_SetAttributeValue;
  CK_C_FindObjectsInit C_FindObjectsInit;
  CK_C_FindObjects C_FindObjects;
  CK_C_FindObjectsFinal C_FindObjectsFinal;
  CK_C_EncryptInit C_EncryptInit;
```

CK_C_Encrypt C_Encrypt;
CK_C_EncryptUpdate C_EncryptUpdate;
CK_C_EncryptFinal C_EncryptFinal;
CK_C_DecryptInit C_DecryptInit;
CK_C_Decrypt C_Decrypt;
CK_C_DecryptUpdate C_DecryptUpdate;
CK_C_DecryptFinal C_DecryptFinal;
CK_C_DigestInit C_DigestInit;
CK_C_Digest C_Digest;
CK_C_DigestUpdate C_DigestUpdate;
CK_C_DigestKey C_DigestKey;
CK_C_DigestFinal C_DigestFinal;
CK_C_SignInit C_SignInit;
CK_C_Sign C_Sign;
CK_C_SignUpdate C_SignUpdate;
CK_C_SignFinal C_SignFinal;
CK_C_SignRecoverInit C_SignRecoverInit;
CK_C_SignRecover C_SignRecover;
CK_C_VerifyInit C_VerifyInit;
CK_C_Verify C_Verify;
CK_C_VerifyUpdate C_VerifyUpdate;
CK_C_VerifyFinal C_VerifyFinal;
CK_C_VerifyRecoverInit C_VerifyRecoverInit;
CK_C_VerifyRecover C_VerifyRecover;
CK_C_DigestEncryptUpdate C_DigestEncryptUpdate;
CK_C_DecryptDigestUpdate C_DecryptDigestUpdate;
CK_C_SignEncryptUpdate C_SignEncryptUpdate;
CK_C_DecryptVerifyUpdate C_DecryptVerifyUpdate;
CK_C_GenerateKey C_GenerateKey;
CK_C_GenerateKeyPair C_GenerateKeyPair;
CK_C_WrapKey C_WrapKey;
CK_C_UnwrapKey C_UnwrapKey;
CK_C_DeriveKey C_DeriveKey;
CK_C_SeedRandom C_SeedRandom;
CK_C_GenerateRandom C_GenerateRandom;
CK_C_GetFunctionStatus C_GetFunctionStatus;

```
  CK_C_CancelFunction  C_CancelFunction;
  CK_C_WaitForSlotEvent  C_WaitForSlotEvent;
};


typedef ck_rv_t (*ck_createmutex_t) (void **mutex);
typedef ck_rv_t (*ck_destroymutex_t) (void *mutex);
typedef ck_rv_t (*ck_lockmutex_t) (void *mutex);
typedef ck_rv_t (*ck_unlockmutex_t) (void *mutex);


struct ck_c_initialize_args
{
  ck_createmutex_t create_mutex;
  ck_destroymutex_t destroy_mutex;
  ck_lockmutex_t lock_mutex;
  ck_unlockmutex_t unlock_mutex;
  ck_flags_t flags;
  void *reserved;
};


#define CKF_LIBRARY_CANT_CREATE_OS_THREADS   (1UL << 0)
#define CKF_OS_LOCKING_OK        (1UL << 1)

#define CKR_OK              (0UL)
#define CKR_CANCEL          (1UL)
#define CKR_HOST_MEMORY          (2UL)
#define CKR_SLOT_ID_INVALID       (3UL)
#define CKR_GENERAL_ERROR       (5UL)
#define CKR_FUNCTION_FAILED       (6UL)
#define CKR_ARGUMENTS_BAD       (7UL)
#define CKR_NO_EVENT          (8UL)
#define CKR_NEED_TO_CREATE_THREADS     (9UL)
#define CKR_CANT_LOCK          (0xaUL)
#define CKR_ATTRIBUTE_READ_ONLY        (0x10UL)
```

```
#define CKR_ATTRIBUTE_SENSITIVE        (0x11UL)
#define CKR_ATTRIBUTE_TYPE_INVALID     (0x12UL)
#define CKR_ATTRIBUTE_VALUE_INVALID    (0x13UL)
#define CKR_DATA_INVALID               (0x20UL)
#define CKR_DATA_LEN_RANGE             (0x21UL)
#define CKR_DEVICE_ERROR               (0x30UL)
#define CKR_DEVICE_MEMORY              (0x31UL)
#define CKR_DEVICE_REMOVED             (0x32UL)
#define CKR_ENCRYPTED_DATA_INVALID     (0x40UL)
#define CKR_ENCRYPTED_DATA_LEN_RANGE   (0x41UL)
#define CKR_FUNCTION_CANCELED          (0x50UL)
#define CKR_FUNCTION_NOT_PARALLEL      (0x51UL)
#define CKR_FUNCTION_NOT_SUPPORTED     (0x54UL)
#define CKR_KEY_HANDLE_INVALID         (0x60UL)
#define CKR_KEY_SIZE_RANGE             (0x62UL)
#define CKR_KEY_TYPE_INCONSISTENT      (0x63UL)
#define CKR_KEY_NOT_NEEDED             (0x64UL)
#define CKR_KEY_CHANGED                (0x65UL)
#define CKR_KEY_NEEDED                 (0x66UL)
#define CKR_KEY_INDIGESTIBLE           (0x67UL)
#define CKR_KEY_FUNCTION_NOT_PERMITTED (0x68UL)
#define CKR_KEY_NOT_WRAPPABLE          (0x69UL)
#define CKR_KEY_UNEXTRACTABLE          (0x6aUL)
#define CKR_MECHANISM_INVALID          (0x70UL)
#define CKR_MECHANISM_PARAM_INVALID    (0x71UL)
#define CKR_OBJECT_HANDLE_INVALID      (0x82UL)
#define CKR_OPERATION_ACTIVE           (0x90UL)
#define CKR_OPERATION_NOT_INITIALIZED  (0x91UL)
#define CKR_PIN_INCORRECT              (0xa0UL)
#define CKR_PIN_INVALID                (0xa1UL)
#define CKR_PIN_LEN_RANGE              (0xa2UL)
#define CKR_PIN_EXPIRED                (0xa3UL)
#define CKR_PIN_LOCKED                 (0xa4UL)
#define CKR_SESSION_CLOSED             (0xb0UL)
#define CKR_SESSION_COUNT              (0xb1UL)
#define CKR_SESSION_HANDLE_INVALID     (0xb3UL)
```

```
#define CKR_SESSION_PARALLEL_NOT_SUPPORTED  (0xb4UL)
#define CKR_SESSION_READ_ONLY       (0xb5UL)
#define CKR_SESSION_EXISTS       (0xb6UL)
#define CKR_SESSION_READ_ONLY_EXISTS     (0xb7UL)
#define CKR_SESSION_READ_WRITE_SO_EXISTS  (0xb8UL)
#define CKR_SIGNATURE_INVALID       (0xc0UL)
#define CKR_SIGNATURE_LEN_RANGE       (0xc1UL)
#define CKR_TEMPLATE_INCOMPLETE       (0xd0UL)
#define CKR_TEMPLATE_INCONSISTENT     (0xd1UL)
#define CKR_TOKEN_NOT_PRESENT       (0xe0UL)
#define CKR_TOKEN_NOT_RECOGNIZED     (0xe1UL)
#define CKR_TOKEN_WRITE_PROTECTED     (0xe2UL)
#define  CKR_UNWRAPPING_KEY_HANDLE_INVALID  (0xf0UL)
#define CKR_UNWRAPPING_KEY_SIZE_RANGE     (0xf1UL)
#define CKR_UNWRAPPING_KEY_TYPE_INCONSISTENT  (0xf2UL)
#define CKR_USER_ALREADY_LOGGED_IN     (0x100UL)
#define CKR_USER_NOT_LOGGED_IN       (0x101UL)
#define CKR_USER_PIN_NOT_INITIALIZED     (0x102UL)
#define CKR_USER_TYPE_INVALID       (0x103UL)
#define CKR_USER_ANOTHER_ALREADY_LOGGED_IN  (0x104UL)
#define CKR_USER_TOO_MANY_TYPES       (0x105UL)
#define CKR_WRAPPED_KEY_INVALID       (0x110UL)
#define CKR_WRAPPED_KEY_LEN_RANGE     (0x112UL)
#define CKR_WRAPPING_KEY_HANDLE_INVALID     (0x113UL)
#define CKR_WRAPPING_KEY_SIZE_RANGE     (0x114UL)
#define CKR_WRAPPING_KEY_TYPE_INCONSISTENT  (0x115UL)
#define CKR_RANDOM_SEED_NOT_SUPPORTED     (0x120UL)
#define CKR_RANDOM_NO_RNG       (0x121UL)
#define CKR_DOMAIN_PARAMS_INVALID     (0x130UL)
#define CKR_BUFFER_TOO_SMALL       (0x150UL)
#define CKR_SAVED_STATE_INVALID       (0x160UL)
#define CKR_INFORMATION_SENSITIVE     (0x170UL)
#define CKR_STATE_UNSAVEABLE       (0x180UL)
#define CKR_CRYPTOKI_NOT_INITIALIZED     (0x190UL)
#define CKR_CRYPTOKI_ALREADY_INITIALIZED  (0x191UL)
#define CKR_MUTEX_BAD          (0x1a0UL)
```

```
#define CKR_MUTEX_NOT_LOCKED        (0x1a1UL)
#define CKR_FUNCTION_REJECTED       (0x200UL)
#define CKR_VENDOR_DEFINED          (1UL << 31)


/* Compatibility layer.  */

#ifdef CRYPTOKI_COMPAT

#undef CK_DEFINE_FUNCTION
#define CK_DEFINE_FUNCTION(retval, name) retval CK_SPEC name

/* For NULL.  */
#include <stddef.h>

typedef unsigned char CK_BYTE;
typedef unsigned char CK_CHAR;
typedef unsigned char CK_UTF8CHAR;
typedef unsigned char CK_BBOOL;
typedef unsigned long int CK_ULONG;
typedef long int CK_LONG;
typedef CK_BYTE *CK_BYTE_PTR;
typedef CK_CHAR *CK_CHAR_PTR;
typedef CK_UTF8CHAR *CK_UTF8CHAR_PTR;
typedef CK_ULONG *CK_ULONG_PTR;
typedef void *CK_VOID_PTR;
typedef void **CK_VOID_PTR_PTR;
#define CK_FALSE 0
#define CK_TRUE 1
#ifndef CK_DISABLE_TRUE_FALSE
#ifndef FALSE
#define FALSE 0
#endif
#ifndef TRUE
#define TRUE 1
#endif
```

```
#endif

typedef struct ck_version CK_VERSION;
typedef struct ck_version *CK_VERSION_PTR;

typedef struct ck_info CK_INFO;
typedef struct ck_info *CK_INFO_PTR;

typedef ck_slot_id_t *CK_SLOT_ID_PTR;

typedef struct ck_slot_info CK_SLOT_INFO;
typedef struct ck_slot_info *CK_SLOT_INFO_PTR;

typedef struct ck_token_info CK_TOKEN_INFO;
typedef struct ck_token_info *CK_TOKEN_INFO_PTR;

typedef ck_session_handle_t *CK_SESSION_HANDLE_PTR;

typedef struct ck_session_info CK_SESSION_INFO;
typedef struct ck_session_info *CK_SESSION_INFO_PTR;

typedef ck_object_handle_t *CK_OBJECT_HANDLE_PTR;

typedef ck_object_class_t *CK_OBJECT_CLASS_PTR;

typedef struct ck_attribute CK_ATTRIBUTE;
typedef struct ck_attribute *CK_ATTRIBUTE_PTR;

typedef struct ck_date CK_DATE;
typedef struct ck_date *CK_DATE_PTR;

typedef ck_mechanism_type_t *CK_MECHANISM_TYPE_PTR;

typedef ck_rsa_pkcs_mgf_type_t *CK_RSA_PKCS_MGF_TYPE_PTR;

typedef struct ck_mechanism CK_MECHANISM;
```

```
typedef struct ck_mechanism *CK_MECHANISM_PTR;

typedef struct ck_mechanism_info CK_MECHANISM_INFO;
typedef struct ck_mechanism_info *CK_MECHANISM_INFO_PTR;

typedef struct ck_function_list CK_FUNCTION_LIST;
typedef struct ck_function_list *CK_FUNCTION_LIST_PTR;
typedef struct ck_function_list **CK_FUNCTION_LIST_PTR_PTR;

typedef struct ck_c_initialize_args CK_C_INITIALIZE_ARGS;
typedef struct ck_c_initialize_args *CK_C_INITIALIZE_ARGS_PTR;

#define NULL_PTR NULL

/* Delete the helper macros defined at the top of the file.  */
#undef ck_flags_t
#undef ck_version

#undef ck_info
#undef cryptoki_version
#undef manufacturer_id
#undef library_description
#undef library_version

#undef ck_notification_t
#undef ck_slot_id_t

#undef ck_slot_info
#undef slot_description
#undef hardware_version
#undef firmware_version

#undef ck_token_info
#undef serial_number
#undef max_session_count
#undef session_count
```

```
#undef max_rw_session_count
#undef rw_session_count
#undef max_pin_len
#undef min_pin_len
#undef total_public_memory
#undef free_public_memory
#undef total_private_memory
#undef free_private_memory
#undef utc_time

#undef ck_session_handle_t
#undef ck_user_type_t
#undef ck_state_t

#undef ck_session_info
#undef slot_id
#undef device_error

#undef ck_object_handle_t
#undef ck_object_class_t
#undef ck_hw_feature_type_t
#undef ck_key_type_t
#undef ck_certificate_type_t
#undef ck_attribute_type_t

#undef ck_attribute
#undef value
#undef value_len

#undef ck_date

#undef ck_mechanism_type_t

#undef ck_rsa_pkcs_mgf_type_t

#undef ck_mechanism
```

```
#undef parameter
#undef parameter_len

#undef ck_mechanism_info
#undef min_key_size
#undef max_key_size

#undef ck_rv_t
#undef ck_notify_t

#undef ck_function_list

#undef ck_createmutex_t
#undef ck_destroymutex_t
#undef ck_lockmutex_t
#undef ck_unlockmutex_t

#undef ck_c_initialize_args
#undef create_mutex
#undef destroy_mutex
#undef lock_mutex
#undef unlock_mutex
#undef reserved

#endif   /* CRYPTOKI_COMPAT */

/* System dependencies.   */
#if defined(_WIN32) || defined(CRYPTOKI_FORCE_WIN32)
#pragma pack(pop, cryptoki)
#endif

#if defined(__cplusplus)
}
#endif

#endif   /* PKCS11_H */
```

・＜Crypto API の場合＞

・サンプルプログラムの概要

CryptoAPI インタフェースの一部を用いたサンプルプログラムです。JAHIS HPKI 対応 IC カードガイドライン Ver.3.0 に従い、作成しています。合わせて参照してください。

サンプルプログラムの利用方法は、コマンドプロンプトにて下記のコマンドを実行することで利用できます。

HPKISignSampleCrypto <CSP_Type> <実行処理> <PIN>

・パラメータの説明
＜CSP_Type＞： 署名用 CSP の場合” sign”、認証用 CSP の場合” auth”
＜実行処理＞：証明書取得の場合” getcert”、 署名の場合：”sign”
＜PIN＞：HPKI カードの PIN

Crypto API インタフェースを利用する際は、以下のプロバイダ名を使用します。

- 電子署名用：HPKI Crypto Service Provider for Non Repudiation
- 電子認証用：HPKI Crypto Service Provider for Authentication

・Crypto API の相互運用性のための機能

HPKI では、Crypto API での相互運用性のために、下表の機能の一覧が実装されることになっています。これらの機能を利用することで、HPKI カードにアクセスし、証明書を使った処理を行うことができます。

（JAHIS HPKI 対応 IC カードガイドライン Ver.3.0　5.2.3. Crypto API インタフェース　表 5 Crypto API より引用）

No	API 名	概要
1	CryptAcquireContext	鍵コンテナのハンドルを生成する
2	CryptReleaseContext	鍵コンテナのハンドルを解放する。
3	CryptGetProvParam	CSP のパラメータの値を取得する。
4	CryptSetProvParam	CSP のパラメータの値を設定する。
5	CryptDestroyKey	鍵の破棄を行う。
6	CryptGetKeyParam	鍵のパラメータの値を取得する。
7	CryptGetUserKey	鍵コンテナ内の鍵ハンドルを取得する。
8	CryptCreateHash	ハッシュオブジェクトの生成を行う。
9	CryptDestroyHash	ハッシュオブジェクトの破棄を行う。
10	CryptSetHashParam	ハッシュオブジェクトのパラメータを設定する。
11	CryptSignHash	ハッシュ値に署名を行う。

それぞれの関数の詳細については、HPKI 対応 IC カードガイドライン Ver. 3 0 の「5.2.3. Crypto API インタフェース」を参照してください。

・サンプルプログラムでの処理の流れ

No.	処理概要	処理内容	CryptoAPI の利用
1	証明書取得	暗号プロバイダとコンテナを指定して、プロバイダハンドルを取得する。	CryptAcquireContext
2		コンテナの保持する RSA 鍵ペアのハンドルを取得する。	CryptGetUserKey
3		鍵ハンドルに付随する利用者証明書データのデータ長を取得する。	CryptGetKeyParam
4		鍵ハンドルに付随する利用者証明書データのデータを取得する。	CryptGetKeyParam
5		鍵ハンドルを破棄する。	CryptDestroyKey
6		プロバイダハンドルを解放する。	CryptReleaseContext
7	署名処理	暗号プロバイダとコンテナを指定して、プロバイダハンドルを取得する。	CryptAcquireContext
8		コンテナの保持する RSA 鍵ペアのハンドルを取得する。	CryptGetUserKey
9		鍵ハンドルに付随する利用者証明書データのデータ長を取得する。	CryptGetKeyParam
10		鍵ハンドルに付随する利用者証明書データのデータを取得する。	CryptGetKeyParam
11		ハッシュオブジェクトの生成	CryptCreateHash
12		署名対象データのハッシュ値を生成（本サンプルではダミーデータ使用)	
13		ハッシュオブジェクトのパラメータを設定	CryptSetHashParam
14		ハッシュ値に署名を行った結果のデータ長を取得する。	CryptSignHash
15		署名用鍵の PIN を設定する。	CryptSetProvParam
16		ハッシュ値に署名を行った結果を取得する。	CryptSignHash
17		ハッシュオブジェクト破棄	CryptDestroyHash
18		鍵ハンドルを破棄する。	CryptDestroyKe
19		プロバイダハンドルを解放する。	CryptReleaseContext

ソースコード＜CryptoAPI インタフェース利用＞

・HPKISignSampleCrypto.c

```
/*---------------------------------------------------------------------*/
/* FILE NAME  :  HPKISignSampleCrypto.c       */
/* VERSION    :  1.0                          */
/* DATE       :  2020/10/29                   */
/*---------------------------------------------------------------------*/

/*=====================================================================*/
/*                              INCLUDE                                */
/*=====================================================================*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <tchar.h>
#include <Windows.h>
#include <wincrypt.h>

/*=====================================================================*/
/*                              DEFINITION                             */
/*=====================================================================*/

#define APROVNAME TEXT("HPKI Crypto Service Provider for Authentication")
#define SPROVNAME TEXT("HPKI Crypto Service Provider for Non Repudiation")
#define HASHSIZE_SHA1 (160 / 8)

/*=====================================================================*/
/*                DEFINITIOIN OF PRIVATE FUNCTION                      */
/*=====================================================================*/

/* プログラムの実行形式　*/
void PrintUsage() {
  printf("Usage : HPKISignSample <prov_type> <method> <pin>¥n");
  printf("prov_type:  auth | sign¥n");
  printf("method:  getcert | sign¥n");
  printf("pin:  HPKICard PIN¥n");
}

/* エラー発生時の処理 */
void printErrorAndExit(const char *f)
{
  DWORD errorcode;
  char errmsg[256];

  errorcode = GetLastError();
  FormatMessageA(
    FORMAT_MESSAGE_FROM_SYSTEM | FORMAT_MESSAGE_IGNORE_INSE
RTS |
```

```
      FORMAT_MESSAGE_MAX_WIDTH_MASK,
      NULL,
      errorcode,
      MAKELANGID(LANG_ENGLISH, SUBLANG_DEFAULT),
      errmsg, sizeof(errmsg) / sizeof(TCHAR),
      NULL);
    printf("%s: 0x%08x (%s).¥n", f, errorcode, errmsg);

    exit(EXIT_FAILURE);
}

/* 各データ(証明書データ、署名データ)を 16 進数表示で標準出力 */
void printHex(const BYTE* data, DWORD len, BOOL limit)
{
    DWORD displen;

    if (limit && len > 256)
        displen = 256;
    else
        displen = len;
    for (DWORD i = 0; i < displen; i++) {
        printf("%02x", *(data + i));
        if (i % 16 == 15 || i == displen - 1)
            printf("¥n");
        else
            printf(" ");
    }
    if (displen < len)
        printf("...¥n");
}

/*
 * getCert: 利用者証明書取得処理
 * 以下の手順で証明書を取得する。
 * 1. プロバイダハンドル取得
 * 2. RSA 鍵ペアのハンドル取得
 * 3. 証明書データ取得
 * 4. 鍵ハンドル破棄
 * 5. プロバイダハンドル解放
 */

void getCert(LPCTSTR pProvName)
{
    HCRYPTPROV hProv;
    HCRYPTKEY  hKey;
    BYTE* certData;
    DWORD certLen;

    /* (1) 暗号プロバイダとコンテナを指定して、プロバイダハンドルを取得する。 */
    if (!CryptAcquireContext(&hProv, NULL, pProvName, PROV_RSA_FULL, 0)) {
        printErrorAndExit("CryptAcquireContext");
```

```
    }

    /* (2) コンテナの保持する RSA 鍵ペアのハンドルを取得する。 */
    if (!CryptGetUserKey(hProv, AT_SIGNATURE, &hKey)) {
      printErrorAndExit("CryptGetUserKey");
    }

    /* (3) 鍵ハンドルに付随する利用者証明書データのデータ長を取得する。 */
    certLen = 0;
    if (!CryptGetKeyParam(hKey, KP_CERTIFICATE, NULL, &certLen, 0)) {
      printErrorAndExit("CryptGetKeyParam(1)");
    }

    certData = malloc(certLen);
    if (certData == NULL) {
      printf("Not enough memory.¥n");
      exit(EXIT_FAILURE);
    }

    /* (4) 鍵ハンドルに付随する利用者証明書データのデータを取得する。 */
    if (!CryptGetKeyParam(hKey, KP_CERTIFICATE, certData, &certLen, 0)) {
      printErrorAndExit("CryptGetKeyParam(2)");
    }

    printf("Certificate:¥n");
    printHex(certData, certLen, FALSE);

    free(certData);

    /* (5) 鍵ハンドルを破棄する。 */
    if (!CryptDestroyKey(hKey)) {
      printErrorAndExit("CryptDestroyKey");
    }

    /* (6) プロバイダハンドルを解放する。 */
    if (!CryptReleaseContext(hProv, 0)) {
      printErrorAndExit("CryptReleaseContext");
    }
}

/*
 * sign: 署名処理
 * 以下の手順で署名を実施する。
 *   1. プロバイダハンドル取得
 *   2. RSA 鍵ペアのハンドル取得
 *   3. 証明書データ取得
 *   4. ハッシュオブジェクト生成
 *   5. ハッシュ値設定
 *   6. 署名データ長取得
 *   7. PIN 設定
```

```
 *  8. 署名データ取得
 *  9. ハッシュオブジェクト破棄
 * 10. 鍵ハンドル破棄
 * 11. プロバイダハンドル解放
 */

void sign(LPCTSTR pProvName, const BYTE *pPinData, DWORD pinLen)
{
  HCRYPTPROV hProv;
  HCRYPTKEY  hKey;
  HCRYPTHASH hHash;
  BYTE* certData;
  DWORD certLen;
  BYTE hashData[256];
  BYTE* sigData;
  DWORD sigLen;

  /* (7) 暗号プロバイダとコンテナを指定して、プロバイダハンドルを取得する。 */
  if (!CryptAcquireContext(&hProv, NULL, pProvName, PROV_RSA_FULL, 0)) {
    printErrorAndExit("CryptAcquireContext");
  }

  /* (8) コンテナの保持する RSA 鍵ペアのハンドルを取得する。 */
  if (!CryptGetUserKey(hProv, AT_SIGNATURE, &hKey)) {
    printErrorAndExit("CryptGetUserKey");
  }

  /* (9) 鍵ハンドルに付随する利用者証明書データのデータ長を取得する。 */
  certLen = 0;
  if (!CryptGetKeyParam(hKey, KP_CERTIFICATE, NULL, &certLen, 0)) {
    printErrorAndExit("CryptGetKeyParam(1)");
  }

  certData = malloc(certLen);
  if (certData == NULL) {
    printf("Not enough memory.¥n");
    exit(EXIT_FAILURE);
  }

  /* (10) 鍵ハンドルに付随する利用者証明書データのデータを取得する。 */
  if (!CryptGetKeyParam(hKey, KP_CERTIFICATE, certData, &certLen, 0)) {
    printErrorAndExit("CryptGetKeyParam(2)");
  }

  printf("Certificate:¥n");
  printHex(certData, certLen, TRUE);

  free(certData);

  /* (11) ハッシュオブジェクトの生成 */
  if (!CryptCreateHash(hProv, CALG_SHA1, 0, 0, &hHash)) {
```

```
    printErrorAndExit("CryptCreateHash");
}

/* (12) 署名対象データのハッシュ値を生成 (本サンプルではダミーデータ使用) */
memset(hashData, 0xa5, HASHSIZE_SHA1);

/* (13) ハッシュオブジェクトのパラメータを設定 */
if (!CryptSetHashParam(hHash, HP_HASHVAL, hashData, 0)) {
    printErrorAndExit("CryptSetHashParam");
}

/* (14) ハッシュ値に署名を行った結果のデータ長を取得する。 */
sigLen = 0;
if (!CryptSignHash(hHash, AT_SIGNATURE, NULL, 0, NULL, &sigLen)) {
    printErrorAndExit("CryptSignHash(1)");
}

/* (15) 署名用鍵の PIN を設定する。 */
if (!CryptSetProvParam(hProv, PP_SIGNATURE_PIN, pPinData, pinLen)) {
    printErrorAndExit("CryptSetProvParam");
}

sigData = malloc(sigLen);
if (sigData == NULL) {
    printf("Not enough memory.¥n");
    exit(EXIT_FAILURE);
}

/* (16) ハッシュ値に署名を行った結果を取得する。 */
if (!CryptSignHash(hHash, AT_SIGNATURE, NULL, 0, sigData, &sigLen)) {
    printErrorAndExit("CryptSignHash(2)");
}

printf("Signature:¥n");
printHex(sigData, sigLen, FALSE);

free(sigData);

/* (17) ハッシュオブジェクト破棄 */
if (!CryptDestroyHash(hHash)) {
    printErrorAndExit("CryptDestroyHash");
}

/* (18) 鍵ハンドルを破棄する。 */
if (!CryptDestroyKey(hKey)) {
    printErrorAndExit("CryptDestroyKey");
}

/* (19) プロバイダハンドルを解放する。 */
if (!CryptReleaseContext(hProv, 0)) {
    printErrorAndExit("CryptReleaseContext");
```

```
  }
}

/*================================================================*/
/*                        Start of main()                         */
/*================================================================*/

int main(int argc, char *argv[])
{
  LPCTSTR pProvName;
  BYTE* pPin;
  DWORD pinLen;

  /*-------------------------------------------------------------------------*/
  /* (A) 引数のチェック                                                      */
  /*-------------------------------------------------------------------------*/

  if (argc != 4)
  {
    printf("実行形式が正しくありません。¥n");
    PrintUsage();
    exit(EXIT_SUCCESS);
  }

  /* CSP 確認
   * auth ： 電子認証用 CSP
   * sign ： 電子署名用 CSP
  */
  switch (*argv[1]) {
  case 'a':
  case 'A':
    pProvName = APROVNAME;
    break;
  case 's':
  case 'S':
    pProvName = SPROVNAME;
    break;
  default:
    printf("実行パラメータ<prov_type>が正しくありません。¥n");
    PrintUsage();
    exit(EXIT_FAILURE);
    break;
  }

  /* PIN */
  pPin = argv[3];
  pinLen = strlen(argv[3]);

  /*-------------------------------------------------------------------------*/
  /* (B) 処理の実行                                                          */
  /*-------------------------------------------------------------------------*/
```

```
    switch (*argv[2]) {
      /* 証明書取得 */
    case 'g':
    case 'G':
      printf("証明書取得処理 開始¥n");
      getCert(pProvName);
      printf("¥n 証明書取得処理 成功¥n");
      break;
      /* 署名 */
    case 's':
    case 'S':
      printf("署名処理 開始¥n");
      sign(pProvName, pPin, pinLen);
      printf("¥n 署名処理 成功¥n");
      break;
    default:
      printf("実行パラメータ<method>が正しくありません。¥n");
      PrintUsage();
      exit(EXIT_FAILURE);
      break;
    }

    exit(EXIT_SUCCESS);
}
/*====================================================================*/
/*                End of main()                                       */
/*====================================================================*/
```

参考文献

＊JAHIS 標準 18-001「JAHIS HPKI 対応 IC カードガイドライン Ver.3.0」2018 年 5 月
https://www.jahis.jp/files/user/04_JAHIS%20standard/18-001_JAHIS%20HPKI%E5%AF%BE%E5%BF%9CIC%E3%82%AB%E3%83%BC%E3%83%89%E3%82%AC%E3%82%A4%E3%83%89%E3%83%A9%E3%82%A4%E3%83%B3Ver.3.0.pdf

問い合わせ先
一般財団法人医療情報システム開発センター
医療情報利活用推進部門
メール：hpki-ad@medis.or.jp

発行

〒162-0825
東京都新宿区神楽坂一丁目 1 番地
電話　03-3267-1922
一般財団法人医療情報システム開発センター
医療情報利活用推進部門